NAVAL POSTGRADUATE SCHOOL

MONTEREY, CALIFORNIA

DISSERTATION

POSE AND WIND ESTIMATION FOR AUTONOMOUS PARAFOILS

by

Charles W. Hewgley IV

September 2014

Dissertation Supervisor: Roberto Cristi

Approved for public release; distribution is unlimited

THIS PAGE INTENTIONALLY LEFT BLANK

REPORT DOCUMENTATION PAGE

Form Approved OMB No. 0704–0188

1. AGENCY USE ONLY *(Leave Blank)*	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	12–09–2014	Doctoral Dissertation 2008-08-13—2013-09-30

4. TITLE AND SUBTITLE

POSE AND WIND ESTIMATION FOR AUTONOMOUS PARAFOILS

5. FUNDING NUMBERS

6. AUTHOR(S)

Charles W. Hewgley IV

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Naval Postgraduate School
Monterey, CA 93943

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Department of the Navy

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

The views expressed in this thesis are those of the author and do not reflect the official policy or position of the Department of Defense or the U.S. Government. **IRB Protocol Number: N/A**

12a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited

12b. DISTRIBUTION CODE

13. ABSTRACT *(maximum 200 words)*

This dissertation presents two contributions to the development of autonomous aerial delivery systems (ADSs), both of which advance the prospect of enabling an ADS to land on a moving platform, such as the deck of a ship at sea. The first contribution addresses the problem of estimating the target's position and velocity. A novel, dual-rate estimation algorithm based on Unscented Kalman filtering allows the ADS to use visual measurements from a fixed monocular sensor to estimate the target's motion even when the ADS's swinging motion in flight causes the target to be out of view. The second contribution addresses the problem of planning a landing trajectory considering winds in the vertical air mass between the target's height and the ADS's altitude. A wind model that assumes a logarithmic relationship between horizontal wind velocity and height in the air mass enables the ADS's guidance algorithm to plan a valid landing trajectory in the presence of these winds. This dissertation contains simulation results for the visual estimation algorithm that show that estimation errors are minimal after estimator convergence. Flight test results indicate that the wind modeling algorithm was useful for computing landing trajectories.

14. SUBJECT TERMS	15. NUMBER OF PAGES
Aerospace and electronic systems, Robot vision systems, Kalman filters, State estimation, Aircraft navigation	245
	16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT	18. SECURITY CLASSIFICATION OF THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
Unclassified	Unclassified	Unclassified	UU

THIS PAGE INTENTIONALLY LEFT BLANK

Approved for public release; distribution is unlimited

POSE AND WIND ESTIMATION FOR AUTONOMOUS PARAFOILS

Charles W. Hewgley IV
Commander, United States Navy
B.S., Carnegie Mellon University, 1992
M.S., Naval Postgraduate School, 2001

Submitted in partial fulfillment of the
requirements for the degree of

DOCTOR OF PHILOSOPHY IN ELECTRICAL ENGINEERING

from the

NAVAL POSTGRADUATE SCHOOL
September 2014

Author: Charles W. Hewgley IV

Approved by:

Roberto Cristi
Professor of Electrical and Computer Engineering
Dissertation Committee Chair and Dissertation Co-Advisor

Oleg A. Yakimenko
Professor of Mechanical and Aerospace Engineering
Professor of Systems Engineering
Dissertation Co-Advisor

Xiaoping Yun
Distinguished Professor of Electrical and Computer Engineering

Robert G. Hutchins
Associate Professor of Electrical and Computer Engineering

Nathan J. Slegers
Associate Professor of Mechanical and Aerospace Engineering

Approved by: R. Clark Robertson
Chair, Department of Electrical and Computer Engineering

Approved by: O. Douglas Moses
Vice Provost for Academic Affairs

THIS PAGE INTENTIONALLY LEFT BLANK

ABSTRACT

This dissertation presents two contributions to the development of autonomous aerial delivery systems (ADSs), both of which advance the prospect of enabling an ADS to land on a moving platform, such as the deck of a ship at sea. The first contribution addresses the problem of estimating the target's position and velocity. A novel, dual-rate estimation algorithm based on Unscented Kalman filtering allows the ADS to use visual measurements from a fixed monocular sensor to estimate the target's motion even when the ADS's swinging motion in flight causes the target to be out of view. The second contribution addresses the problem of planning a landing trajectory considering winds in the vertical air mass between the target's height and the ADS's altitude. A wind model that assumes a logarithmic relationship between horizontal wind velocity and height in the air mass enables the ADS's guidance algorithm to plan a valid landing trajectory in the presence of these winds. This dissertation contains simulation results for the visual estimation algorithm that show that estimation errors are minimal after estimator convergence. Flight test results indicate that the wind modeling algorithm was useful for computing landing trajectories.

THIS PAGE INTENTIONALLY LEFT BLANK

Table of Contents

1 Introduction: Shipboard Delivery **1**

1.1 Problem Description 1

1.2 Prescriptive Scenario 15

1.3 Contributions of this Work 18

1.4 Dissertation Organization 20

2 Aerial Delivery to the Present Day **21**

2.1 A Short History of Military Aerial Delivery 21

2.2 Modern Aerial Delivery 24

2.3 Beginnings of Maritime Use 25

2.4 Potential for Maritime Use 27

2.5 Current State of the Art in Precision Aerial Delivery 30

2.6 Relevant Research in Other Fields 35

2.7 Relationship to Research in this Dissertation 42

3 Visual Sensing for Terminal Guidance **45**

3.1 Projecting a 3D Scene on a 2D Image Plane 45

3.2 Method for Target Motion Estimation 54

3.3 Summary of the Dual-rate Estimation Algorithm 90

3.4 Simulation of the Kalman Estimator 97

4 Wind Estimation and Wind Profile Modeling **129**

4.1 Understanding the Wind Environment 129

4.2 Wind Estimation and Profile Modeling for Shipboard Landing 135

4.3 Description of the Snowflake ADS Guidance Algorithm 138

4.4 Computing D_{switch} using a Logarithmic Wind Profile 143

4.5 Adaptive Filtering for Parameter Estimation 153

4.6 Wind Estimation Algorithm Implementation 155

4.7 Dynamic Parafoil Model Simulation 160

4.8 Post-processing Flight Test Data 169
4.9 Flight Test Comparison of Wind Profiling Methods 173

5 Conclusions and Future Work **179**
5.1 Summary of Findings 179
5.2 Contributions of this Work 180
5.3 Suggestions for Future Research 180

Appendix A Code Listings **185**
A.1 Van Loan Method for Computing Φ and $\mathbf{Q}$ 185
A.2 Computing D_{switch} Using Logarithmic Wind Profile. 185
A.3 RLS Estimation of Wind Profile Parameters 186
A.4 Processing Snowflake Recorded Data 189
A.5 Archiving Snowflake Data as a MATLAB Structure. 190

Appendix B System Development and Flight Test **193**
B.1 Snowflake System History. 193
B.2 Snowflake Data Recording 194
B.3 Flight Test Chronology 195

List of References **199**

Initial Distribution List **215**

List of Figures

Figure 1.1 Comparison of traditional VERTREP with precision aerial delivery 2

Figure 1.2 Final Turn Trajectory in 3D 3

Figure 1.3 ADS in flight with moving target landing platform 7

Figure 1.4 Landing ADS with wind profile 10

Figure 1.5 Wind profile examples 11

Figure 1.6 Standard shipboard approaches for helicopters 12

Figure 1.7 Trajectory comparison between Onyx and Snowflake 13

Figure 1.8 Two ships with different locations for helicopter landing decks . 14

Figure 1.9 Overview of aerial delivery resupply mission 16

Figure 1.10 Release point for aerial delivery resupply mission 17

Figure 1.11 Aerial delivery system rendezvous with target 18

Figure 2.1 Early conical parachute design by Leonardo da Vinci 22

Figure 2.2 Airdropped supplies near Bastogne, France 23

Figure 2.3 Relief supplies delivered from the air to Haiti 25

Figure 2.4 Aerial delivery of ransom payment to pirates 26

Figure 2.5 Maritime Craft Air Delivery System (MCADS) 27

Figure 2.6 Aerial delivery of an aircraft part 28

Figure 2.7 Aerial delivery payload being retrieved by small boat 29

Figure 2.8 Snowflake autonomous parafoil in flight 32

Figure 2.9 RQ-8A Fire Scout UAS prepares for autonomous landing 38

Figure 3.1 Geometry of the perspective-projective transformation 47

Figure 3.2 Overview of geometry of global and image reference frames . . . 49
Figure 3.3 Target features in global frame and image features on image plane 67
Figure 3.4 Motion estimation subsystem switching diagram 72
Figure 3.5 Two views of the target 74
Figure 3.6 Epipolar geometry 75
Figure 3.7 Epipolar constraint scalar distance 76
Figure 3.8 Epipolar constraint with both image planes 77
Figure 3.9 Two line-of-sight vectors to the target 79
Figure 3.10 Target-centered coordinate reference frame 82
Figure 3.11 Two estimation modes 92
Figure 3.12 Artificial trajectory simulation image plane and overhead 98
Figure 3.13 Geometry of simulation to evaluate epipolar measurement 99
Figure 3.14 Hybrid demonstration simulation raw and processed video 101
Figure 3.15 Hybrid simulation used for estimator performance analysis . . . 103
Figure 3.16 Portion of test flight used for hybrid simulation 105
Figure 3.17 Example of spline fitting to flight test data 106
Figure 3.18 Two examples of noise modeling 107
Figure 3.19 Simple trajectory simulation position estimation errors 109
Figure 3.20 Simple trajectory simulation speed estimation errors 110
Figure 3.21 Simple trajectory simulation course estimation errors 111
Figure 3.22 Simple trajectory simulation estimated target track 115
Figure 3.23 Velocity estimation error after target returns to view 116
Figure 3.24 Velocity estimation error with no noise 117
Figure 3.25 Velocity estimation error with process noise 118

Figure 3.26 Model validation estimation error plots 120
Figure 3.27 Best parameter set simulation estimation errors 122
Figure 3.28 Nominal parameter set simulation estimation errors 124
Figure 3.29 Fast parameter set simulation estimation errors 126
Figure 3.30 Simulation estimation errors with realistic noise 128

Figure 4.1 The wind triangle 131
Figure 4.2 Constant and piece-wise-constant wind profiles 133
Figure 4.3 Logarithmic wind profile 135
Figure 4.4 Simulation of turbulent airflow aft of a ship's superstructure . . . 137
Figure 4.5 Snowflake ADS Guidance strategy overview 139
Figure 4.6 Snowflake ADS terminal guidance maneuver overview 141
Figure 4.7 Sequence of iterative computations for D_{switch} 145
Figure 4.8 RLS adaptive filtering algorithm block diagram 154
Figure 4.9 Landing motion of the ADS in a relative coordinate frame 158
Figure 4.10 Constant wind profile in 6DOF simulation 163
Figure 4.11 Constant wind profile simulation trajectory 164
Figure 4.12 Logarithmic wind profile in 6DOF simulation 165
Figure 4.13 Logarithmic wind profile simulation trajectory 166
Figure 4.14 Final turn parameters L_x and D_{switch} plotted versus time 167
Figure 4.15 Final turn parameters L_x and D_{switch} plotted versus time 167
Figure 4.16 Logarithmic wind profile in 6DOF simulation 169
Figure 4.17 Logarithmic wind profile simulation trajectory 170
Figure 4.18 Constant wind profile compared to logarithmic in simulation . . . 171

Figure 4.19 Parafoil trajectory overview using constant wind profile 172
Figure 4.20 Constant wind profile compared to logarithmic in flight test . . . 173
Figure 4.21 Logarithmic wind profile compared to constant profile in simulation 174
Figure 4.22 Final turn parameter estimates using logarithmic wind model . . 175
Figure 4.23 Parafoil trajectory overview using logarithmic wind profile . . . 176
Figure 4.24 Side-by-side trajectory comparison using constant wind profile . 177

List of Tables

Table 2.1 Snowflake system characteristics 31

Table 3.1 Presumed values of sensor estimation errors 63

Table 3.2 Presumed values of process model errors 64

Table 3.3 Errors in the initial state estimates 64

Table 3.4 Nominal and improved sets of estimator parameters 103

Table 3.5 Position estimation error with long out-of-view duration 112

Table 3.6 Velocity estimation error with long out-of-view duration 113

Table 3.7 Velocity estimation error with high measurement noise 114

Table 4.1 Overall results of simulation trade study 168

Table B.1 Chronological history of Snowflake flight tests 198

THIS PAGE INTENTIONALLY LEFT BLANK

List of Acronyms and Abbreviations

2D	two-dimensional
3D	three-dimensional
6DOF	six degrees of freedom
AAS	Advanced Autopilot System
ABL	atmospheric boundary layer
ADS	aerial delivery system
ADSC	Aerodynamic Decelerator Systems Center
AFRL	Air Force Research Laboratory
AGL	above ground level
AGU	airborne guidance unit
ASW	anti-submarine warfare
AUV	autonomous underwater vehicle
BYU	Brigham Young University
CARP	computed air release point
CCD	charge-coupled device
CFD	computational fluid dynamics
CLF	Combat Logistics Force
DR	dead-reckoning
DZ	dropzone
EADS	European Aeronautic Defence and Space Company

EKF extended Kalman filter

FOB forward operating base

GN&C guidance, navigation, and control

GPS Global Positioning System

GSM Global System for Mobile Communications

GT Georgia Institute of Technology

IDVD Inverse Dynamics in the Virtual Domain

IMU inertial measurement unit

INRIA Institut National de Recherche en Informatique et en Automatique

JCTD Joint Capability Technology Demonstration

JMDSE Joint Medical Distance Support and Evacuation

JPADS Joint Precision Airdrop System

LIDAR light detection and ranging

LOP line of position

MCADS Maritime Craft Air Delivery System

MEMS micro-electro-mechanical systems

MLW Micro Light-Weight

MPC model predictive control

NPS Naval Postgraduate School

NSRDEC Natick Soldier Research, Development, and Engineering Center

ONR Office of Naval Research

PATCAD Precision Airdrop Technology Conference and Demonstration

PID proportional-integral-derivative

RLS recursive least squares

SLAM simultaneous localization and mapping

SPS standard positioning service

TIP Turn Initiation Point

TMA target motion analysis

TNT Tactical Network Topology

TPBVP two-point boundary-value problem

UAH University of Alabama in Huntsville

UAS unmanned aircraft system

UNREP underway replenishment

UKF Unscented Kalman filter

USNA U.S. Naval Academy

USV unmanned surface vehicle

VERTREP vertical replenishment

VGA Video Graphics Array

WOD wind-over-deck

YPG Yuma Proving Ground

THIS PAGE INTENTIONALLY LEFT BLANK

Nomenclature

$\{b\}$	Body-fixed reference frame
$\widehat{\mathbf{C}}_{\mathbf{y}}$	estimated covariance matrix of random vector $\mathbf{y}$
$\widehat{C}_y$	estimated covariance scalar value of random variable y
${}^{\mathrm{g}}\mathbf{d}_o$	displacement of observer frame origin from global frame origin
$\mathbf{\Phi}$	discrete-time state transition matrix (only used in calculation of state error covariance matrix $\mathbf{P}$)
$\mathbf{F}$	continuous-time state matrix which can be the Jacobian matrix of nonlinear function $\mathbf{f}(\cdot)$
$\mathbf{f}(\cdot)$	nonlinear function that calculates time-derivative of state vector $\mathbf{x}$
f	imaging sensor focal length
$\mathbf{g}(\cdot)$	nonlinear function for computing pseudomeasurement from measurement vector $\mathbf{z}$
$\{g\}$	global reference frame
$\mathbf{H}$	measurement matrix for in-view estimation mode
$\mathbf{h}^{\mathrm{T}}$	row vector for calculating predicted scalar measurement $\hat{s}$ (back-in-view mode)
$\mathbf{H}'$	modified measurement matrix incorporating $\mathbf{h}^{\mathrm{T}}$ for use with scalar measurement $\hat{s}$
$\{i\}$	image reference frame
$\mathbf{K}$	gain matrix for state estimate update equation
$\mathbf{\Lambda}$	Euler angle triplet of aerial delivery system
$\mathbf{M}$	matrix for computation of target location from image position
N	number of time steps between target leaving view, and returning to view
$\mathbf{P}$	projection matrix
$\mathbf{P}$	state estimate error covariance matrix
$\mathbf{Q}[n]$	discrete-time process noise covariance matrix
$\mathbf{Q}_c$	continuous-time process noise matrix used to compute $\mathbf{Q}[n]$
$\mathbf{R}(\cdot)$	rotation matrix
$\mathbf{R}$	measurement noise covariance matrix
$\bar{\boldsymbol{\sigma}}$	vector of standard deviations of added noise to raw measurements
$\mathbf{S}$	matrix for computation of target course from image orientation

s	scalar distance measurement used for back-in-view epipolar constraint calculation
$\hat{s}$	predicted value of scalar measurement s
ς	transformed sample for scalar measurement $\hat{s}$
T_s	fundamental sampling interval, assumed to be constant
${}^{\mathrm{i}}u, {}^{\mathrm{i}}v$	coordinates of image on image plane
$(\Delta u, \Delta v)$	target bow and stern feature differences in distance on image plane
$\mathbf{v}$	measurement noise vector
V	target speed (velocity)
$\mathbf{v}_1, \mathbf{v}_2$	lines of sight in two-view geometry formulation
V_{f},V_{r}	downwind (forward) or upwind (reverse) measured velocity of the ADS on a straight flight path
$V_{\mathrm{h}}^{\star}$	steady-state no-wind horizontal velocity of the ADS
V_{T}	speed of the target
$V_{\mathrm{v}}^{\star}$	steady-state no-wind vertical velocity of the ADS
W	sample weighting value for calculating mean or covariance
W,$\widehat{W}$	horizontal wind speed, true value and estimate
w, w_i, w_o, w'	scaling factor for homogeneous coordinates for image, object, and their ratio
$\hat{\mathbf{x}}$	estimate of state vector
${}^{\mathrm{g}}x_T, {}^{\mathrm{g}}y_T$	horizontal coordinates of target in global coordinate frame
x_{T}	position coordinate of the target
$\Delta^{\mathrm{g}}x_{T_f}, \Delta^{\mathrm{g}}y_{T_f}$	difference in target features bow and stern in global coordinates
$\mathbf{y}$	random variable used to calculate mean and covariance of transformed samples
ψ_T	target course
$\mathbf{z}$	measurement vector
ζ	transformed sample vector for in-view measurement
Z	set of measurement vectors z constituting the sigma points (sample vectors) for the Unscented Transform

Executive Summary

Autonomous aerial delivery systems (ADSs) are autonomous vehicles that are deployed from an airborne carrier vehicle and that descend to the earth under a steerable round parachute or rectangular parafoil canopy. These systems are used by the U.S. Army and U.S. Air Force most often for resupply of ground forces. In this way, ADSs represent the modern evolution of aerial delivery, or *airdrop*, techniques that came into widespread use during the Second World War. Two current Micro Light-Weight systems (5 kg to 77 kg) are shown in Figure 1. Image 1a is the Mosquito by Stara Technologies, Inc. and image 1b is the Onyx by Atair Aerospace, Inc. These two images suggest that the various systems currently in use in general have very similar configurations.

Modern ADSs have achieved a high degree of accuracy in guiding toward a fixed target on land. In recent conflicts in Iraq and Afghanistan, the U.S. Air Force and U.S. Army have relied on aerial delivery to resupply forward operating bases that are widely dispersed about the region of operations and often in rough terrain.

The two contributions to the science of aerial delivery contained in this dissertation were developed with the goal of enabling an autonomous ADS to land on a moving platform, such as the landing deck of a ship at sea. To accomplish this feat, the ADS must estimate both the position and velocity of the target and then plan a trajectory to follow down to the target. The first contribution described herein is the development of a novel, dual-rate estimation scheme based on an Unscented Kalman filter that enables a moving observer to use measurements from a monocular visual sensor to estimate the position and velocity of a target even when the target is intermittently out of view. The second contribution is the development of an algorithm that allows a landing ADS to assume a logarithmic functional relationship between height above landing and horizontal wind speed in its calculation of landing trajectory. Key details for each of these contributions follow below.

An overhead depiction of an ADS in flight estimating target position and velocity to compute an intercept point is illustrated in Figure 2. The main challenge of the visual estimation portion of the landing task is that the typical ADS tends to oscillate in yaw in flight. This oscillation can be caused by wind or by the vehicle dynamics after a control actuation.

(a) Mosquito ADS (b) Onyx ADS

Figure 1: Two aerial delivery systems (ADSs) in flight. Both are Micro Light-Weight systems: 5 kg (a) and 77 kg (b) (images reproduced from Lafond *et al.* [1].)

Rather than adding a complicated panning mount to the visual sensor, an algorithm was developed that uses the motion of the ADS in an advantageous way. When the target swings out of the fixed field of view of the visual sensor, and then later swings back in view, these two snapshots of the target are used to generate one measurement based on the two-view epipolar constraint. The duration that the target remains out of view represents one sampling interval, and a much shorter interval (higher sampling rate) is used when the target remains in view. Furthermore, the relationship between the three-dimensional (3D) state of the target and the two-dimensional (2D) measurement of its image on the image plane is highly nonlinear. The estimation algorithm described in this dissertation uses the recently-developed Unscented Transform to characterize errors in the target estimate based on errors in measurement.

A profile view of an ADS's planned landing trajectory to the aft landing deck of the ship is depicted in Figure 3. To plan a landing trajectory such that the ADS's arrival at the height of the landing deck is simultaneous with its arrival at the intercept location, the guidance algorithm must consider the effect of the wind throughout the entire descent. A sample profile of horizontal wind velocity versus height is shown on the left side of Figure 3. Because the ADS will have no measurements of wind velocity at altitudes below its own, it needs a working assumption, or model, of wind horizontal velocities between

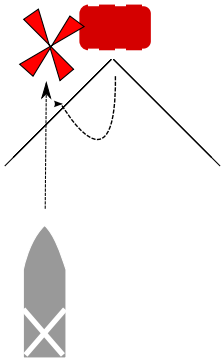


Figure 2: Estimating target position and intercept. An ADS in flight uses a sensor to estimate target motion and to compute an intercept point in this overhead depiction.

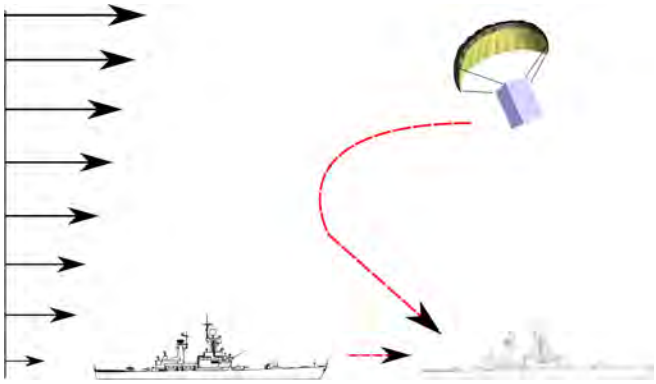


Figure 3: Trajectory for shipboard landing. An ADS in flight plans a trajectory for shipboard landing considering an estimated wind profile.

its altitude and the landing height. A logarithmic model of near-surface boundary layer winds represents a good balance between fidelity and simplicity. To take advantage of the logarithmic profile, a prototype ADS was developed that used an algorithm for estimating the shape of the logarithmic wind profile based on current altitude winds. The wind profile parameters so determined were also incorporated into the landing trajectory calculations.

A development team consisting of members from Naval Postgraduate School (NPS) and University of Alabama in Huntsville built a small prototype ADS called *Snowflake* and

Figure 4: Snowflake prototype ADS. The Snowflake prototype ADS glides toward landing at Camp Roberts, California.

conducted a series of flight tests starting in 2008 and continuing to the present day. During some of these tests, the team gathered data to support visual estimation algorithm development; during others, the team evaluated an implementation of the wind modeling algorithm. The Snowflake prototype is shown in Figure 4 in flight above McMillan Airfield, Camp Roberts, California, which was the location for the majority of flight tests. Video data gathered from a camera mounted to the Snowflake during the flight tests served as a model upon which a simulation of the visual estimation algorithm was developed. The estimation algorithm was implemented in the Simulink modeling environment and was executed using synthetic video input.

The output of the simulations indicated that the estimation algorithm was able to converge to small position estimation error values in times on the order of 20 s to 30 s under various measurement noise conditions. The algorithm was also found effective in estimation of target velocity. Convergence to small values of estimation error occurred within 15 s under various measurement noise conditions. The wind modeling algorithm was flight tested in simultaneous side-by-side deployments comprising one system using the logarithmic wind profile model and one system using the simple assumption of a constant wind profile. The system using the logarithmic wind profile was found to make correct choices in the terminal phase of flight. Foremost among the recommendations for future research is the integration into the Snowflake prototype of a visual sensing system that records video using a time reference common to all telemetry and other sensor data.

Reference

[1] K. Lafond, R. Benney, M. Henry, A. Meloni, C. Ormonde, G. Noetscher, S. Patel, M. Shurtliff, S. Tavan, A. Goldenstein, and G. Pinnell, "Joint medical distance support and evacuation joint capability technology demonstration," in *21st AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar*, American Institute of Aeronautics and Astronautics. Dublin, Ireland: American Institute of Aeronautics and Astronautics, 23–26 May 2011. [Online]. Available: http://dx.doi.org/10.2514/6.2011-2574

THIS PAGE INTENTIONALLY LEFT BLANK

Acknowledgments

I feel truly privileged that I was able to learn from each one of my dissertation committee members: in the classroom and in the field. To professors Roberto Cristi, Oleg Yakimenko, Xiaoping Yun, Gary Hutchins, and Nathan Slegers, you have my profound thanks! Also, I am grateful to the field testing team for Snowflake: Oleg Yakimenko, Nathan Slegers, Eugene Bourakov, and the expert ground team from Arcturus UAV: "Red" Jensen, Drew Robinson, Josh Malone, and Phil Heidt. You all proved that field testing is the best engineering classroom there is! Thanks are also due to the leadership at Arcturus UAV, D'Milo Hallerberg and Erik Folkestad, who were enthusiastic about our crazy idea when they first heard it.

A special debt of gratitude is of course due to my dissertation co-advisors, Roberto Cristi and Oleg Yakimenko. To say that it has been an adventure getting to this point would be an understatement! Certainly, I made frequent excursions from the path, but I thank you both for driving me on to the goal.

Finally, the biggest debt is owed to my wife, Maria, and my daughters, Saga and Disa. None of us knew how hard this would be, and you certainly shouldered your share of the burden. There is more to say than can be put into words.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 1:
Introduction: Shipboard Delivery

The resupply of ships at sea is a challenging task that is a hallmark of U.S. Navy blue-water operations. One common method that naval forces use to transfer supplies from one ship to another is known as vertical replenishment (VERTREP). This technique involves a rotary-wing aircraft transferring sling loads from the deck of one ship to the deck of another. While it is both effective and fast, this method is also risky and expensive. This chapter introduces autonomous aerial delivery systems (ADSs) as a means to provide a fast but safe and economical method of delivering cargo to a ship underway. The chapter opens with a mathematical analysis of the problem of landing an ADS on a moving platform such as a ship and continues with a sample operational scenario in which an ADS would be used for such a task.

1.1 Problem Description

A typical ADS is shown in Figure 1.1b. The U.S. Army at its Natick Soldier Research, Development, and Engineering Center (NSRDEC), in Natick, Massachusetts, is actively developing such systems under the Joint Precision Airdrop System (JPADS) program [1]. Shown is the *Ultrafly* by Wamore, Inc., which is designed to deliver a suspended cargo pallet to a fixed location on the ground. The system has the capability of self-steering in flight by using control lines to deflect the left and right trailing edges of the parafoil canopy shown above the suspended cargo. The Ultrafly ADS uses an on-board Global Positioning System (GPS) receiver to determine its current position and then to compute guidance commands to a preprogrammed fixed location on the ground. The current shipboard delivery technique using a piloted helicopter is shown in Figure 1.1a. For shipboard delivery capability, the first and most obvious necessary change to an autonomous ADS such as the Ultrafly is to replace the guidance algorithm with one designed to seek a moving target. In this section, the mathematical framework is constructed for the problem of terminal guidance of an autonomous parafoil to a moving landing platform.

(a) (b)

Figure 1.1: Comparison of traditional VERTREP with precision aerial delivery. USS *Sacramento* (AOE-1) in the foreground performs VERTREP with USS *Lassen* (DDG-82) in the background in (a). U.S. Navy photograph by Photographer's Mate Airman Nicole Carter, 2003. In (b), the Ultrafly ADS is shown in flight. Image reproduced from Lafond *et al.* from [2].

1.1.1 Trajectory Planning and Optimization

An oft-repeated maxim for large projects is to "begin with the end in mind." For the study of ADSs, this idea suggests analyzing the landing problem first. The final outcome, or final conditions, of the landing problem are simple: the ADS should arrive at the coordinates of the target, and at the surface of the world, simultaneously. This desired final condition lends itself to optimization techniques: finding a solution—a final trajectory—that minimizes the differences between desired final conditions and achieved final conditions. This section contains an example of one optimization technique developed jointly by Naval Postgraduate School (NPS) and University of Alabama in Huntsville (UAH) during ADS research and testing. The purpose of describing this example is to show how the key problem aspects to be discussed subsequently, target estimation and wind profile modeling, fit within the larger framework of the landing problem.

Consider the three-dimensional (3D) depiction of the landing trajectory shown in Figure 1.2. The final trajectory is shown as a thick dashed line between the two terminal points. The ADS must conform to certain conditions at these boundary points; therefore, the tra-

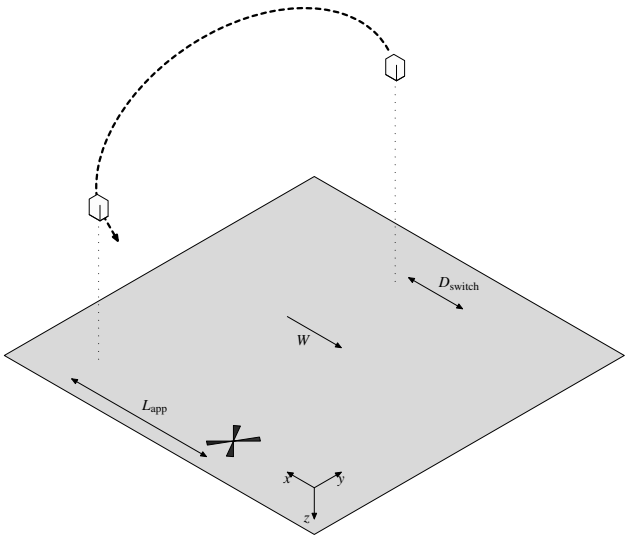


Figure 1.2: Final Turn Trajectory in 3D. The intended landing target of the ADS is shown as an X on the world surface.

jectory planning problem is related to the two-point boundary-value problem (TPBVP) family. In this case, the second terminal point is not actually the target on the surface but a point in the air at the beginning of the final straight-in descent to the target. The TPBVP is solved to determine what path the ADS should steer to travel from an initial state $\mathbf{x}_0$, the first boundary point, to a final state $\mathbf{x}_f$, the second boundary point, arriving at the final state at a specified final time t_f.

These boundary points consist of two horizontal position coordinates and vehicle track and are summarized as:

$$\mathbf{x}_0 = \begin{bmatrix} x_0 \\ y_0 \\ \psi_0 \end{bmatrix} \text{ at } t_0 \qquad \mathbf{x}_f = \begin{bmatrix} x_f \\ y_f \\ \psi_f \end{bmatrix} \text{ at } t_f \tag{1.1}$$

where ψ is defined as the angle between the ADS's track over the ground and the positive x-axis. The vertical coordinate is missing from the boundary points because the final trajectory is planned in two dimensions only; the ADS descends at a constant rate in the vertical dimension. In summary, the optimization problem is to calculate a sequence of steering commands $\psi(t)$ such that the ADS travels from $\mathbf{x}_0$ to $\mathbf{x}_f$, minimizing the difference between $\mathbf{x}_f$ and the actual final state achieved, at the time of arrival t_f.

The optimization technique used by the NPS and UAH researchers to solve this TPBVP is a simple but powerful and flexible technique known as Inverse Dynamics in the Virtual Domain (IDVD). This technique is introduced by Yakimenko [3], and its application to the guidance of the Pegasus parafoil is described in subsequent articles with Kaminer [4] and with Kaminer and Pascoal [5]. The use of IDVD for the specific purpose of calculating an optimal trajectory for an autonomous parafoil is well documented by Slegers and Yakimenko [6], [7], and will only be summarized here. With this method, a virtual parameter $\bar{\tau} = \tau/\tau_f$ is used as a parameter for polynomials for both dimensions of the two-dimensional planar trajectory. The assumption of constant vertical velocity is used to account for the third dimension of parafoil motion. The parametrization of the two-dimensional planar trajectory in the x- and y-directions of the coordinate system are parametrized as two polynomials $P_1(\bar{\tau})$ and $P_2(\bar{\tau})$ as

$$\begin{aligned} x(\bar{\tau}) &= P_1(\bar{\tau}) = a_{10} + a_{11}\bar{\tau} + a_{12}\bar{\tau}^2 + a_{13}\bar{\tau}^3 + b_{11}\sin(\pi\bar{\tau}) + b_{12}\sin(2\pi\bar{\tau}) \\ y(\bar{\tau}) &= P_2(\bar{\tau}) = a_{20} + a_{21}\bar{\tau} + a_{22}\bar{\tau}^2 + a_{23}\bar{\tau}^3 + b_{21}\sin(\pi\bar{\tau}) + b_{22}\sin(2\pi\bar{\tau}) \end{aligned} \tag{1.2}$$

where coefficients a_{ij} and b_{ij} are defined by initial and final boundary conditions on polynomials $P_1(\bar{\tau})$ and $P_2(\bar{\tau})$.

One of the hallmarks of the IDVD method is that, in this method, the virtual parameter $\bar{\tau}$ is *not* directly proportional to time. In fact, the correspondence between the virtual domain and the time domain is accomplished by defining a *speed factor* λ such that

$$\lambda(\tau) = \frac{\mathrm{d}\tau}{\mathrm{d}t}. \tag{1.3}$$

The decoupling of the virtual domain and the time domain in IDVD allows the shape of the trajectory to be varied independently of the trajectory's duration in time. In other words, the trajectory's shape can be independent of the speed at which the parafoil moves along the trajectory.

To apply the IDVD technique to the moving target landing problem, the boundary conditions first need to be specified. The initial conditions $\mathbf{x}_0$ are the elements of the current state of the ADS at the instant that it initiates the final turn; the final turn is then triggered by a separate process. The final conditions $\mathbf{x}_f$ are that the ADS is airborne at some distance

L_{app} directly in line with the target intercept position along the assumed wind vector and that the ADS is tracking directly at the intercept position. The desired straight-in approach time $T_{\text{app}}^{\text{des}}$, along with the ADS's no-wind horizontal speed $V_{\text{h}}^{\star}$ and horizontal wind speed W, determine the approach distance:

$$L_{\text{app}} = (V_{\text{h}}^{\star} + W) T_{\text{app}}^{\text{des}}. \tag{1.4}$$

This value combines with the computed position of the target at the landing time (the target intercept position) Equation (1.7) to constitute the full set of final conditions:

$$\mathbf{x}_f = \begin{bmatrix} x_{\text{T}}(t_{\text{start}}) - V_{\text{T}} \frac{z_{\text{start}}}{V_{\text{v}}^{\star}} + (V_{\text{h}}^{\star} + W) T_{\text{app}}^{\text{des}} \\ 0 \\ -\pi \end{bmatrix} \text{ at } t = t_f, \tau = \tau_f. \tag{1.5}$$

The final value of the x-coordinate here assumes constant wind profile W. In these equations, t_{start} is the starting time of the final approach maneuver, and x_{T} and V_{T} are the position coordinate and velocity of the target, respectively. The variables related to the ADS are its starting altitude z_{start} as well as its no-wind horizontal and vertical velocities $V_{\text{h}}^{\star}$ and $V_{\text{v}}^{\star}$. The horizontal wind speed, assumed to be constant in this case, is denoted by W.

Next is the calculation of the coefficients in the polynomial equation (1.2) and also the derivatives of these equations using the boundary conditions expressed in terms of virtual parameter τ. There is only one varied parameter for this optimization problem, which is the final virtual parameter value τ_f. An efficient gradient-free optimization algorithm can then perform these steps iteratively, seeking to find the value of τ_f that minimizes a specified cost function.

Once the algorithm determines the optimal value of τ_f, it can then define a discrete set of computation points from $\tau = 0$ to $\tau = \tau_f$ and calculate desired track ψ and turn rate $\dot{\psi}$ at each of these points. The set of turn rate values $\dot{\psi}$ are then a schedule of control inputs for the ADS's autopilot to follow. The advantage of having a future schedule of control inputs is that this schedule is useful for advanced control schemes such as model predictive control (MPC). Slegers and Costello have applied this control scheme to parafoil guidance [8].

Another advantage that this method has is that an optimal solution can be easily and quickly recalculated while the ADS is in flight. If unexpected winds deflect the ADS from the optimal trajectory, the algorithm can simply compute another optimal trajectory using the ADS's current state for the initial conditions.

Notice that target estimation and wind profile modeling have a part to play in the final trajectory calculation because both target and wind parameters appear in the final boundary conditions in Equation (1.5). Even though Equation (1.5) has only a single value of W and assumes a constant wind profile, Yakimenko and Slegers [9] have recently extended this formulation to handle various wind profiles.

1.1.2 Target Motion Estimation

Slegers and Yakimenko [7] devised the coordinate framework that is relevant to this terminal guidance problem. Consider the overhead view of the landing scenario shown in Figure 1.3. This is the same landing scenario as that shown in Figure 1.2, this time shown with more details in two dimensions. A diagram of the coordinate frame is shown in the lower-left corner, with the x- and y-axes in the plane of the world's surface and the z-axis positive in the down direction. The ADS's parafoil canopy is shown on the right side, initially moving in the positive x-direction. The target ship is shown in the upper-left corner, initially moving in the negative x-direction. A vector representing prevailing wind direction in the negative x-direction is shown at bottom center.

For the moving target scenario, one must first define the starting time, labeled t_{start}. In this case, it is assumed that the ADS follows a two-stage trajectory of which the first stage is a loitering stage during which the ADS flies a holding pattern upwind of the target while calculating the moment at which it should exit the loiter pattern and begin the approach for landing. Therefore, t_{start} is defined as this moment of exiting the loiter pattern. In Figure 1.3, the labels t_{start} indicate the positions of the ADS and the target at this time.

The target's location may be defined by the change in its position along the x-axis from the moment the parafoil leaves the loitering phase to the moment the parafoil lands; that is, from t_{start} until landing. The assumption, in this case, is that the landing platform is moving in the *negative* x-direction with constant speed. The resultant change in x-coordinate, is

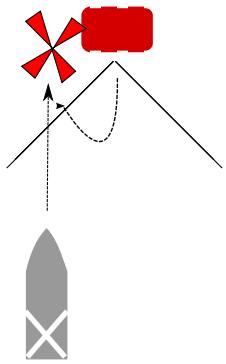


Figure 1.3: ADS in flight with moving target landing platform. In this overhead depiction, the ADS estimates target motion to compute an intercept point.

defined as:

$$\Delta x_{\mathrm{T}} = V_{\mathrm{T}} \frac{z_{\mathrm{end}} - z_{\mathrm{start}}}{V_{\mathrm{v}}^{\star}} = -V_{\mathrm{T}} \frac{z_{\mathrm{start}}}{V_{\mathrm{v}}^{\star}} \tag{1.6}$$

where the ending altitude z_{end} is assumed to be zero. The ratio $z_{\mathrm{start}}/V_{\mathrm{v}}^{\star}$ determines the time duration from the moment the loitering phase is ended to the moment the parafoil lands, and V_{T} is the constant speed of the target. Furthermore, z_{start} is the altitude of the ADS at t_{start}, and $V_{\mathrm{v}}^{\star}$ is an assumed-constant vertical velocity. Note that Equation (1.6) was previously published in a paper by Hewgley [10, Eq. 1]; however, the right-side of the equation was erroneously given as positive in that paper. Equation (1.6) is the corrected version with a negative sign now properly in place before V_{T}. The position of the landing platform at any given moment along the x-axis is labeled $x_{\mathrm{T}}(t)$. This definition is justified because in actual implementation, $x_{\mathrm{T}}(t)$ is a value that is updated on every execution of the guidance algorithm's software control loop.

The guidance algorithm calculates the distance L from the ADS's position at t_{start} to a point abeam of the target's final position at the time of landing along the x-axis. In words, these expressions state that, commencing at t_{start}, the parafoil must move a distance L along the x-axis before commencing the final turn to land on a moving target that is at a position on

the x-axis determined by

$$x_{\mathrm{T}}(t_{\mathrm{end}}) = x_{\mathrm{T}}(t_{\mathrm{start}}) + \Delta x_{\mathrm{T}} = x_{\mathrm{T}}(t_{\mathrm{start}}) - V_{\mathrm{T}} \frac{z_{\mathrm{start}}}{V_{\mathrm{v}}^{\star}}. \tag{1.7}$$

For convenience, let the x-coordinate of the ADS's starting position be zero. Assuming that the target traveled at a constant velocity V_{T} in the negative x-direction from its starting location $x_{\mathrm{T}}(t_{\mathrm{start}})$, we find the value of distance L is

$$L = x_{\mathrm{T}}(t_{\mathrm{start}}) - \frac{V_{\mathrm{T}}}{V_{\mathrm{v}}^{\star}} z_{\mathrm{start}}. \tag{1.8}$$

Slegers and Yakimenko used an on-target landing condition—that the ADS must be both at the target and also at ground level simultaneously—to solve for an expression relating the ADS starting altitude z_{start} to the following parameters [7]:

- $V_{\mathrm{h}}^{\star}$, the ADS's no-wind, constant horizontal velocity,
- $V_{\mathrm{v}}^{\star}$, the ADS's constant vertical velocity,
- T_{turn}, the time duration for the ADS to complete the final 180° turn,
- $T_{\mathrm{app}}^{\mathrm{des}}$, the desired time duration for the ADS's final, straight approach to landing,
- L, the distance that the ADS travels past the landing location before initiating its final turn, and,
- W, assumed constant horizontal wind velocity along the x-axis toward the coordinate frame origin.

Equation (1.8) is useful to recast distance L in terms of parafoil and target motion. The resulting equation is solved for z_{start},

$$z_{\mathrm{start}} = -V_{\mathrm{v}}^{\star} \frac{x_{\mathrm{T}} + V_{\mathrm{h}}^{\star}(T_{\mathrm{turn}} + 2T_{\mathrm{app}}^{\mathrm{des}})}{V_{\mathrm{h}}^{\star} - W - V_{\mathrm{T}}} \tag{1.9}$$

which expresses the altitude at which the ADS must exit the loitering phase in order to achieve a landing on the moving target platform. Equation (1.9) indicates that for the guidance algorithm to calculate z_{start}, it must have estimates of target position x_{T}, target speed V_{T}, and horizontal wind speed W. A key assumption that allowed a simple expression for Equation (1.9) was that the speed of the wind W was constant throughout the descent of the

ADS. In general, the wind varies with height and with time, and so a better mathematical representation of the wind merits further investigation.

1.1.3 Wind Profile Modeling

By their very nature, all unpowered ADSs that rely on parachutes or parafoils are gliding machines and will drift with the wind. When landing accuracy is a performance goal for an ADS, some estimate, or model, of the wind in that part of the atmosphere at and below the ADS's altitude, down to the earth's surface, must be considered. This model takes the form of a wind profile, or a description of the horizontal wind magnitude, and sometimes wind direction, over a range of heights above ground level.

Knowledge of an accurate wind model is important because wind disturbance during the landing phase of an autonomous parafoil's flight is a leading contributor to miss distance. In fact, flight tests of a small prototype ADS conducted by NPS and UAH have indicated that errors due to unknown winds, along with errors in altitude estimation, were the two largest contributors to miss distance [11].

For an autonomous parafoil system, unknown winds can have their most profound effect on landing accuracy just prior to touchdown. For the trajectory illustrated in Figure 1.3, the ADS commences a final 180° turn for landing at the target assuming that this final approach is aligned with the wind vector. Furthermore, Equation (1.9), which determines the altitude z_{start} at which the ADS should exit the loiter phase and begin the approach phase, is based on the simplifying assumption that the final wind estimate is constant down to the target. The assumption of constant wind velocity across a range of altitudes near the earth's surface does not agree with empirical evidence. Introductory meteorology texts such as Salby [12] and Stull [13] examine this topic in detail.

Consider the profile view of the ADS's final landing trajectory to a moving target shown in Figure 1.4. Vectors representing horizontal wind velocity at altitudes from the surface up to the ADS's starting height are shown on the left side of the figure. In Figure 1.4 the horizontal wind velocity increases with altitude, but not according to a simple linear relationship.

One method of defining a wind profile is with an analytic functional relationship, such as

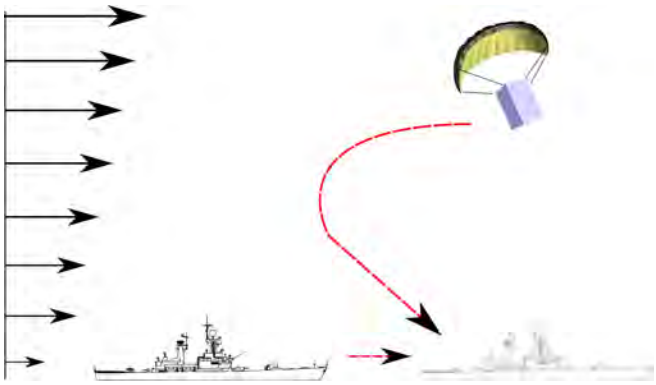


Figure 1.4: Landing ADS with wind profile. An ADS in flight plans a trajectory to landing considering some estimated wind profile.

horizontal wind magnitude being a function of height above the surface, or $W(z)$. If the guidance algorithm has knowledge of what the functional relationship for the wind model is, then it can calculate values of horizontal wind velocity for every intermediate height between its current altitude and the earth's surface and incorporate this knowledge into its trajectory planning. A challenge for the descending ADS in this situation is that it might have no direct measurement of winds below its current altitude. In other words, the ADS may not be able to rely on a ground-based sensor or on another airborne sensor at a lower altitude to transmit wind information.

In this case, the guidance algorithm must make an assumption about the functional relationship $W(z)$, such as in the following examples:

$$W(z) = \begin{cases} W_1 & z < \alpha \\ W_2 & \alpha \leq z < \beta \\ W_3 & \beta \leq z \end{cases} \qquad \text{piecewise constant} \qquad (1.10)$$

$$W(z) = \alpha z + \beta \qquad \text{linear} \qquad (1.11)$$

$$W(z) = \alpha \ln(-z) + \beta \qquad \text{logarithmic} \qquad (1.12)$$

where the vertical coordinate z is positive in the down direction from the origin at the surface of the earth, as defined in Figure 1.3; therefore, the value of z for an object in flight is negative. Equations (1.11) and (1.12) demonstrate that, along with the assumption

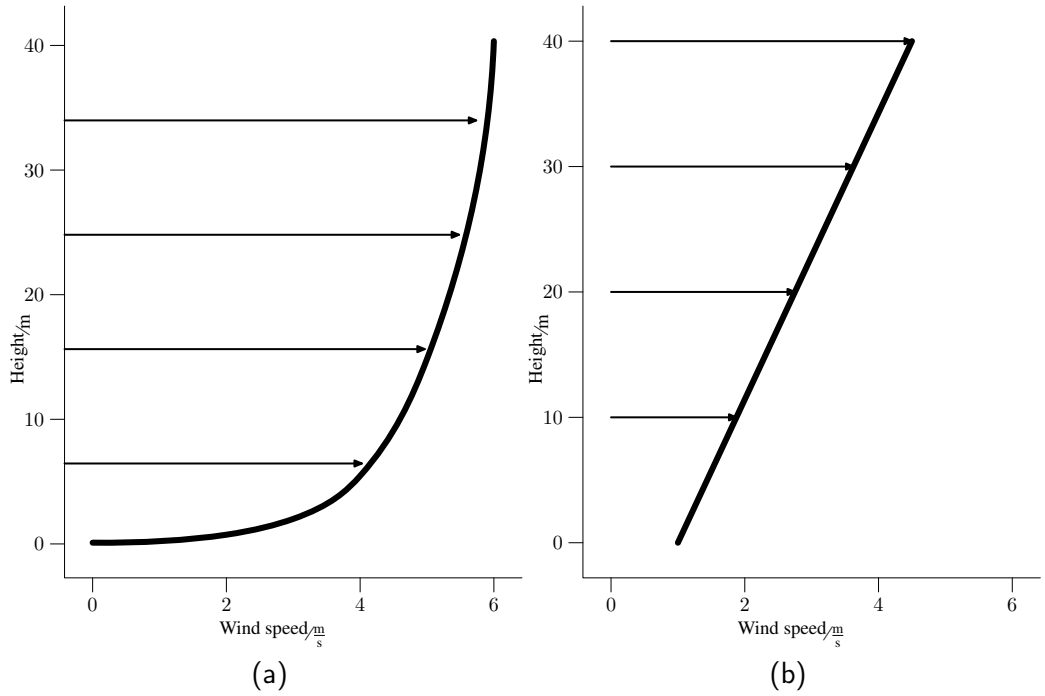


Figure 1.5: Wind profile examples. These include: (a) logarithmic, and (b) linear profiles.

of the form of the functional relationship, there may also be unknown parameters, such as α and β in these examples. A graphical representation of the linear and logarithmic profiles is shown in Figure 1.5. In summary, the wind profile modeling problem for the guidance algorithm is to estimate the unknown parameters of the assumed wind model as the ADS descends through the air mass, collecting horizontal wind measurements at its current altitude. A recursive estimation technique lends itself to this problem.

At this point, the two main aspects of the problem are:

target estimation, determining through some means both the position and velocity of a target moving on a two-dimensional (2D) world surface, and

wind profile modeling, constructing a mathematical functional relationship $W(z)$ for wind velocity in the air mass at all altitudes between the surface target and the airborne vehicle.

These two aspects of the problem, once solved, enable the ADS to plan a final trajectory

for landing on the target. The trajectory planning process is examined in more detail in Section 1.1.1.

1.1.4 Other Considerations

To this point, this section has contained a discussion of the technical aspects of the shipboard landing problem which identified two key aspects of the problem: target estimation, and wind profile modeling. It is fair now to ask whether shipboard delivery using an ADS is feasible. In other words, is shipboard aerial delivery a task that could be accomplished on a regular basis with operational U.S. Navy ships? The answer to that question is unknown because shipboard aerial delivery by an autonomous ADS has not even been demonstrated in an experimental setting to date. The remainder of this section contains a survey of traits and capabilities which may make shipboard landing by an autonomous ADS both possible and feasible.

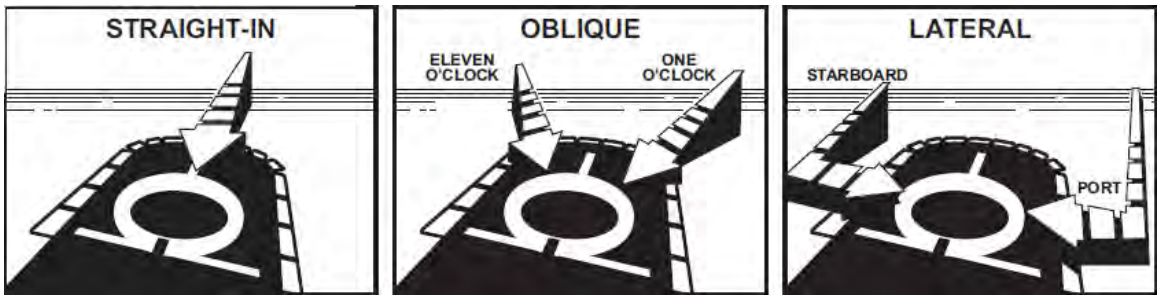
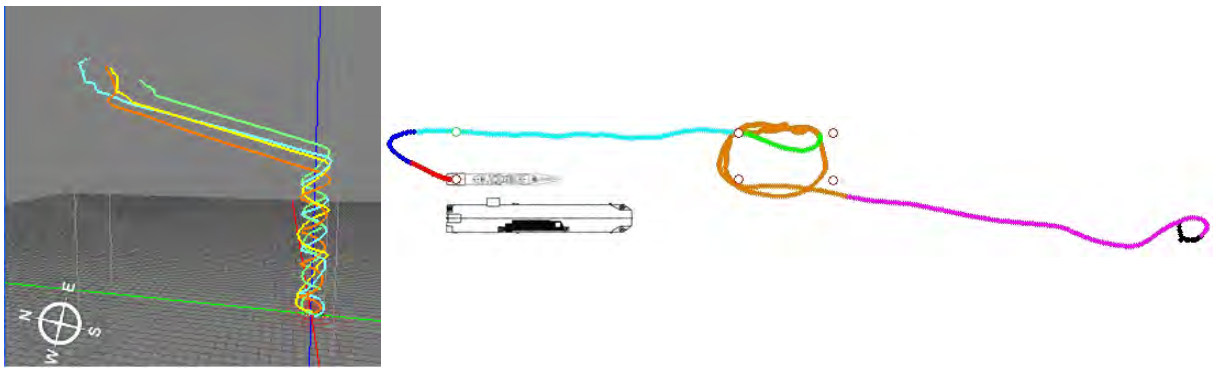


Figure 1.6: Standard shipboard approaches for helicopters. These approaches are from a standard procedures manual for flight deck operations from [14].

One place to start is to examine helicopter shipboard landing, which is practiced and executed with great regularity by the U.S. Navy. Various standard helicopter approach paths are shown in Figure 1.6. In these methods, the helicopter approaches the landing deck from astern, aft of the ship's superstructure. For an ADS to replicate such an approach, it must have the capability to plan and execute an appropriate final trajectory.

Consider the approach to landing trajectories by two different ADSs shown in Figure 1.7. The spiraling approach trajectory flown by a set of four Onyx MLW ADSs developed by Atair Aerospace, Inc. is shown in Figure 1.7a, and a recorded flight trajectory of the NPS and UAH Snowflake prototype ADS is shown in Figure 1.7b. The Snowflake trajectory is shown adjacent to an overhead image of a helicopter carrier as a notional illustration of a



(a) (b)

Figure 1.7: Trajectory comparison between Onyx and Snowflake. The descent trajectory of a set of four Onyx MLW ADSs is shown in 3D [15] in (a). For comparison, a notional Snowflake ADS trajectory is shown in (b).

shipboard approach. The Snowflake flies a box-shaped loiter pattern, then a straight path to a 180° final turn to landing. To replicate a standard helicopter approach, the planned approach of the Snowflake ADS is preferable to the simple, spiraling approach of the Onyx ADS because a spiral approach from overhead entails a high risk of collision with the ship's superstructure.

Another enhancement that may be necessary for feasibly landing a parafoil accurately on a moving target is the capability to aim for a specific location or landing area on the target itself. This capability is useful because different ships have different landing areas as a function of the ship's design. For example, U.S. Navy combatant ships are configured to have the flight deck located on the fantail aft, whereas some auxiliary ships may have the appropriate landing area on a forward deck, toward the bow of the ship. A few ships, notably the hospital ships USNS *Mercy* (T-AH-19) and USNS *Comfort* (T-AH-20), have helicopter flight decks positioned amidships, as illustrated in Figure 1.8. The capability for the ADS to select a specific area on the target for landing suggests the use of a visual sensor both for terminal guidance and for landing area identification.

In the scenario of logistic resupply to a vessel underway proposed in the opening of this chapter, the target is *cooperative*. In other words, the target maintains a course and speed most conducive to a successful landing by the ADS. The target could also be cooperative

Figure 1.8: Two ships with different locations for helicopter landing decks. USNS *Pecos* (T-AO-197) in the foreground and USNS *Mercy* (T-AH-19) in the background demonstrate various configurations for helicopter landing deck placement. Official U.S. Navy photograph by Chief Photographer's Mate E. G. Martens.

by having an automatic beacon that broadcasts the target's position, course, and speed at regular intervals so the landing algorithm has more information with which to plan its trajectory. On the other hand, for ADSs in general to be useful, they should not require the installation of significant additional equipment on board the destination ships.

Another possible scenario is that the target vessel is *unaware* of the presence of the airborne ADS and, therefore, is *uncooperative*. This scenario could involve the effort to land a covert sensor package aboard a vessel of interest. In this case, the target, being unaware, is most likely not maneuvering evasively but neither is it broadcasting signals that are especially helpful to the ADS. The uncooperative target that *is* maneuvering evasively for some reason is another matter. In that case, the ADS would almost certainly fail in its landing attempt. For the ship to evade the landing ADS is exceedingly easy; therefore, the uncooperative and evasive target scenario is not considered in this work.

1.2 Prescriptive Scenario

A logical extension to the feasibility discussion in Section 1.1.4 is to question whether shipboard delivery using an ADS is practical. In other words, does shipboard aerial delivery make military or economic sense? That question is not answered in this dissertation; however, a first step in constructing an answer is to describe a scenario in which shipboard aerial delivery might be used.

1.2.1 Mission Planning

The traditional method of resupplying ships underway is known as underway replenishment (UNREP). This method involves sending a support ship out to rendezvous with the ship to be resupplied and then maneuvering both ships in a side-by-side formation while stores are transferred using cables and fuel is transferred through hoses connecting the two ships. The two vessels in Figure 1.8 are engaged in UNREP. During an UNREP, a VERTREP can also occur; in fact, the two ships need to be fairly close to make VERTREP feasible. The key advantage of aerial delivery over UNREP and VERTREP is speed of response: an aerial delivery mission could conceivably be planned and executed in a matter of hours, not days as in the case of traditional UNREP. A shore-to-ship helicopter VERTREP is possible but requires that the receiving ship be fairly close to shore. For these reasons, aerial delivery to ships underway, far from the coast, would be especially useful for high-value, mission-critical components for which at-sea spares are not readily available, such as components for ever-more sophisticated shipborne radars and unmanned aircraft systems (UASs).

Mission planning for aerial delivery resupply of high-value components to a ship underway in the South China Sea might proceed as follows. A surface combatant underway on individual patrol is shown in Figure 1.9. At this patrol location, an aerial delivery mission can be planned and executed in hours, whereas it may take days for the needed component to be loaded aboard a Combat Logistics Force (CLF) ship and for the CLF ship to be diverted to rendezvous with the combatant. A cargo aircraft such as a C-130 Hercules transport is launched instead to rendezvous with the ship.

1.2.2 Rendezvous and System Deployment

Upon arriving at the ship's location, the aircraft's crew calculates coordinates of the location upwind of the ship's position at which the ADSs will be deployed from the aircraft.

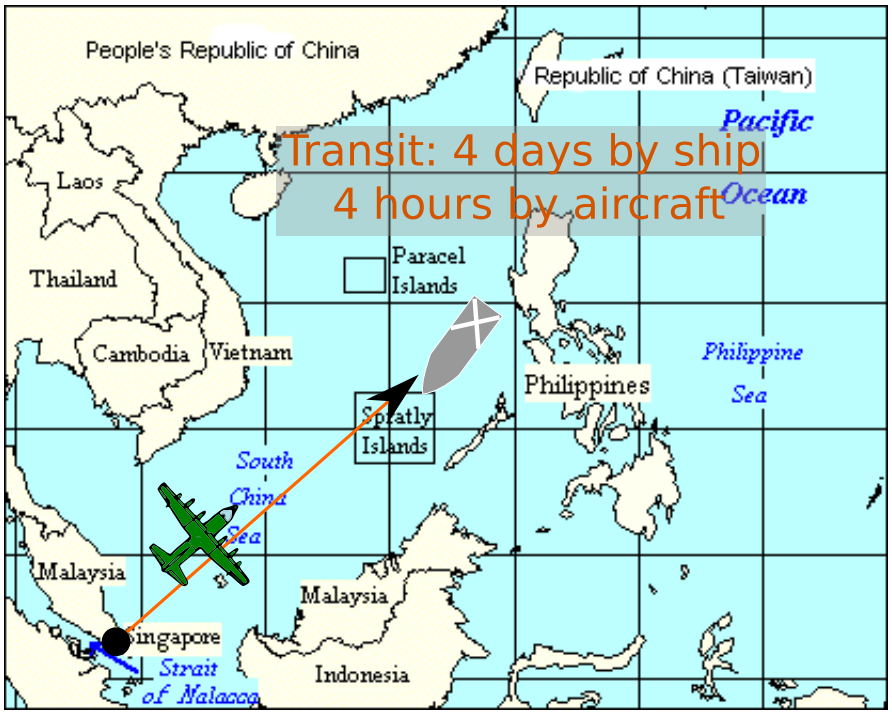


Figure 1.9: Overview of aerial delivery resupply mission. The combatant ship on patrol in this case is best served with aerial delivery resupply. Map Source: U.S. Energy Information Administration, International Hydrographic Organization.

This point is called the computed air release point (CARP) by definition in the U.S. Department of Defense dictionary of terms [16]. An overhead view of the CARP is shown in Figure 1.10. This point is calculated such that, with the prevailing winds and the receiving ship's motion, the ADS is able to calculate a feasible trajectory that it will be able to fly to a landing on the ship's aft flight deck.

1.2.3 Aerial Delivery System Approach

After one or more ADSs are deployed, each ADS enters a holding pattern as it descends and calculates its approach to the target ship. In this scenario, the receiving ship is a cooperative target; however, if the receiving ship wishes to maintain radio silence, a visual sensor on the ADS can track the landing area on the receiving ship and provide guidance commands to the ADS autopilot. The calculation of the final portion of the trajectory to landing is, of course, the most critical phase of the flight. The trajectory planning algorithm aboard the ADS must be sophisticated enough to plan a precise landing trajectory that will approach

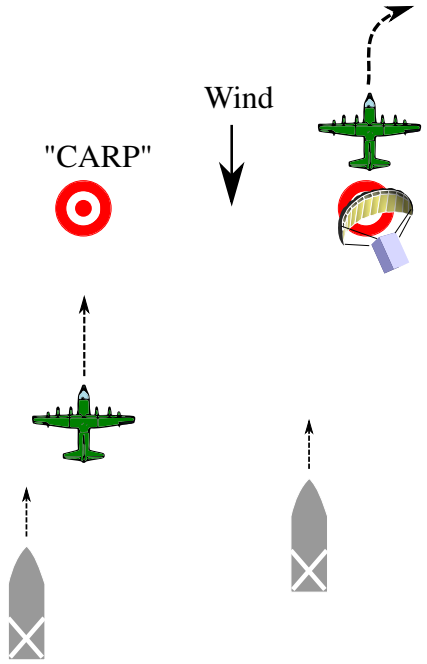


Figure 1.10: Release point for aerial delivery resupply mission. The transport aircraft computes the CARP in flight, then releases the ADS at that point.

the receiving ship's flight deck from aft of the ship's superstructure. The holding pattern and transition to the final trajectory is shown in Figure 1.11.

1.2.4 Contingency Design Features

Future design evolutions of the ADS described above must consider contingencies such as the ADS being unable to reach the ship for an on-deck landing or that the ship's captain deems the approach of the ADS unsafe and commands an abort. For these instances, provision must be made for emergency flotation in the container system so that the ADS can make a landing at sea in the vicinity of the receiving ship. The cargo container must

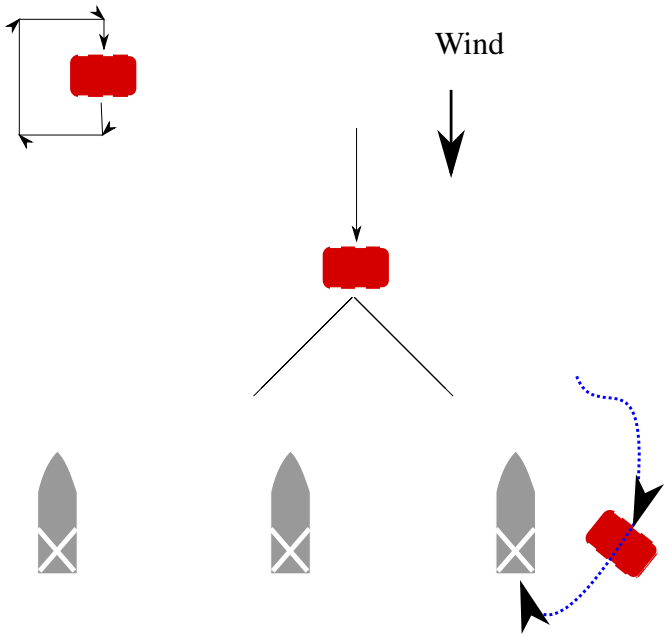


Figure 1.11: Aerial delivery system rendezvous with target. The ADS flies a holding pattern, then acquires the target and computes a trajectory to land.

have the capability to be hoisted on deck from the sea surface. This requirement sets the maximum weight limit for an individual cargo container.

1.2.5 Military Utility of Shipboard Aerial Delivery

The concept of operations for the use of aerial delivery for the resupply of ships underway has significant potential in the near term as an additional logistic delivery method that can be used when an extremely quick response time is needed. As future mission demands on the U.S. Navy's fleet grow, while the capacity of the CLF simultaneously declines, the use of aerial delivery may become a necessity for future logistics planners.

1.3 Contributions of this Work

So far in this chapter, the technical fundamentals of the shipboard landing problem for an autonomous parafoil have been introduced, and a first examination has been made of the feasibility of cargo deliveries to U.S. Navy ships using this idea. The contributions to the science of aerial delivery that are contained in the later technical chapters of this work are detailed in this section.

The main aim of this work is to advance the science of autonomous aerial delivery so that it can be useful to the U.S. Navy in a maritime setting. The idea of maritime use of ADSs for cargo delivery was first published by Hewgley [17]; the aim of this dissertation is to advance the science of autonomous ADSs for this purpose. Toward this aim, the two main contributions of this work are a target estimation method for an airborne ADS to determine the target's course and speed using visual sensing and a wind profile modeling method that enables the ADS to calculate the effects of winds on its landing trajectory.

The target estimation method developed in this dissertation uses a monocular vision sensor. Such a sensor is passive and does not require any supporting equipment to be installed on the target. Monocular sensors are inexpensive to implement for rapid prototyping and field testing. The target estimation method features a novel, dual-rate estimation scheme: the estimator uses a constant sampling rate when the target is continuously in view and a variable sampling rate that handles periods when the target is out of view by sampling just the scenes immediately before the target disappears and immediately after the target reappears. This technique is necessary because the target intermittently leaves the field of view of the fixed monocular sensor due to the inherent side-to-side swinging motion of an ADS in flight. The target estimation algorithm is based on Unscented Kalman estimation with two different underlying state-space models: one model when the target is in view and another model for when the target has been out of view and then returns to view. The Unscented Transform is used as part of this algorithm to estimate uncertainty in the inherently nonlinear transformation from the 3D scene to the 2D image.

The wind profile modeling method presented in this dissertation offers an improvement to the simple assumption of constant or piecewise constant horizontal wind velocity profile found in the original implementation of the Snowflake ADS guidance algorithm by Slegers and Yakimenko [7]. A method for including the assumption that the horizontal wind profile is not constant but follows a logarithmic curve of wind speed versus height above ground is described in this dissertation. The wind profile modeling algorithm estimates the shape of the logarithmic curve using a recursive least squares (RLS) estimation scheme with successive wind measurements made by the ADS as it descends. This wind model more accurately represents typical wind profiles in nature and better allows the ADS's guidance algorithm to calculate the effect the wind will have on the ADS's landing trajectory. This

wind model was evaluated during flight tests of the Snowflake prototype ADS in which the logarithmic wind profile model enabled Snowflake to calculate the point at which to initiate the final turn to the target. The results of this series of flight test experiments involving the logarithmic wind model have been published in a paper by Hewgley and Yakimenko [18].

In fact, flight testing was a large part of the effort involved in this dissertation work. A secondary contribution of this dissertation is a set of scripts written in the MATLAB programming language for processing and storing in data structures recorded flight test data from several versions of the Snowflake prototype ADS.

1.4 Dissertation Organization

Following this chapter, a brief history of the military use of aerial delivery is contained in Chapter 2 to provide a perspective on the origins of the technology in current ADSs. A summary of current research in ADS technology as well as a survey of other research related to autonomous landing, computer vision, and visual estimation is contained in Chapter 2.

The theoretical development of the visual estimation algorithm, as well as results of a simulation of the working estimation algorithm based on video from a camera externally attached to the Snowflake ADS prototype during flight testing, is contained in Chapter 3.

Chapter 4 begins with an introduction to the atmospheric science upon which the logarithmic wind profile is based, followed by an explanation of how the wind model affects the decision-making of the guidance algorithm used in the Snowflake ADS. Results of flight testing conducted to validate and evaluate the logarithmic wind model versus the constant wind profile assumption are contained in Chapter 4.

The conclusions drawn both from the simulations described in Chapter 3 and from the flight tests detailed in Chapter 4 and Appendix B are summarized in Chapter 5. Also, this final chapter contains suggestions for directions for continuing research.

MATLAB code listings including important routines used for simulation and data processing techniques used to analyze recorded experimental data are contained in Appendix A.

An overview of the flight tests conducted related to the NPS and UAH Snowflake project from 2009 to 2011 is contained in Appendix B.

CHAPTER 2:
Aerial Delivery to the Present Day

Airdrop is considered a fundamental military capability today, and with this capability, modern armies are more mobile and better supplied than ever before. Autonomous ADSs exist today that use sophisticated guidance, navigation, and control (GN&C) systems to fly automatically to a desired target on the surface of the earth after being deployed from the aircraft. This chapter begins with a short history of military airdrop to provide context to the more recent developments in aerial delivery technology. These recent developments are the subject of the end of this chapter, along with a review of other research in the field of autonomous systems relevant to the shipboard landing problem.

2.1 A Short History of Military Aerial Delivery

Airdrop (Also air delivery) "The unloading of personnel or materiel from aircraft in flight" [from Joint Publication 1-02 "Dictionary of Military and Associated Terms"] [16]

The history of air drop, or air delivery, or, more formally, *aerial delivery*,[1] begins with the conception of the parachute, which is the quintessential tool of this trade. One of the first recorded designs of a parachute is credited to none other than that master inventor of the Renaissance, Leonardo da Vinci, who sketched a design for a conical parachute in the fifteenth century [19, p. 42]. It was not until the nineteenth century that a successful jump from a height using a parachute was demonstrated; early balloonists were pioneers of using parachutes as means to escape burning balloons [19, p. 42].

2.1.1 Demonstration of Military Utility

The potential military utility of the parachute and aerial delivery was realized early in the twentieth century by the famous American military aviator Major General William "Billy" Mitchell, U.S. Army. General Mitchell, just prior to the conclusion of World War I, was planning an airdrop of a U.S. Army infantry division behind German lines using large bomber aircraft that had been developed for Britain's Royal Air Force during the war;

[1] Henceforth, the term *aerial delivery* is used in preference to the other related terms—consider the other terms to be synonyms.

Figure 2.1: Early conical parachute design by Leonardo da Vinci. The parachute is one of the many inventions from the mind of the Italian Renaissance genius (1452–1519) that was only realized after his time. This photograph of a sketch in the margin of a da Vinci notebook dates from 1483.

however, the Armistice was signed before these plans could be put into action [20, p. 1]. It was not the Americans, but the Russian Red Army that next took the lead by being first to demonstrate the use of parachute-borne soldiers in the mid-1930s. The Red Army's Airlanding Corps staged demonstrations in which not only soldiers, but also crew-served weapons, were landed, quickly assembled, and moved [21, p. 47].

2.1.2 Wartime Logistics Use

A quotation that is often heard in military settings is "Amateurs talk tactics, professionals talk logistics." With the first use of aerial delivery in World War II, there was much talk of the tactics of employment of airborne forces, but the use of aerial delivery for logistics and resupply led to some remarkable results. Following World War II, aerial delivery for logistics was again used during the conflicts in Korea and Vietnam.

Bastogne, France, 1944

In December, 1944, during the Battle of the Bulge, the *Screaming Eagles* of the famed American 101st Airborne Division were surrounded by German forces near the French town of Bastogne. When asked by German forces to surrender the forces under his command, Brigadier General Anthony McAuliffe famously replied "Nuts!" [19, p. 45] Over the next five days, 1046 tons of ammunition and supplies were delivered using airdrop to the 101st, enabling them to hold out until General George Patton's Fourth Armored Division broke through the German lines [21].

Figure 2.2: Airdropped supplies near Bastogne, France. Soldiers of the 101st Airborne Division retrieve supplies during the Battle of the Bulge in 1944. U.S. Army photograph from Cole [22].

Leyte, Philippines, 1944

In 1942, General Douglas MacArthur withdrew from the Philippines, vowing "...I shall return!" In 1944, he did just that as U.S. Army forces landed at Leyte Island. Another innovation in aerial delivery was introduced by the Eleventh Airborne Division during the battle for this island when components for two portable surgical hospitals were delivered using parachutes into the mountains of Leyte, allowing medical personnel to treat casualties for whom such care would have otherwise been unavailable [21, p. 228].

Chosin Reservoir, Korea, 1950

The use of aerial delivery to supply a surrounded force was again demonstrated in the winter of 1950 when U.S. Marines of the First Marine Division were surrounded by Chinese forces in North Korea at the Chosin Reservoir. During the ensuing two-week battle, 1571 tons of supplies were delivered using airdrop to the Marines, as well as eight sections of an M-2 portable bridge, each weighing 4500 lbs [23]. The Marines were eventually able to break out to the port of Hungnam, where they were taken aboard U.S. Navy ships and evacuated.

Khe Sanh, Vietnam, 1968

Aerial delivery for resupply of ground forces was undertaken on an even greater scale almost two decades later when U.S. Marines of the Third Marine Amphibious Force (III MAF) and elements of the South Vietnamese army at their fortified outpost at Khe Sanh, Vietnam, were besieged by several division-sized elements of the North Vietnamese army. Twice during the siege, the ammunition storage area inside the compound was destroyed by rocket and mortar fire; but, over the 78 days of this battle, 8120 tons of ammunition and supplies airdropped to the compound enabled the Marines and South Vietnamese soldiers to mount a successful defense [23].

2.2 Modern Aerial Delivery

The use of aerial delivery for resupply of military ground forces has not become obsolete; in fact, it is in widespread use during current military campaigns. The dispersed nature of the ground force deployments in both Iraq and Afghanistan make aerial delivery even today a vital logistics tool. In fact, the estimated total weight of delivered goods in Iraq and Afghanistan in 2008 was more than 16 million pounds [1]. Modern use of aerial delivery is also characterized by improvements in technology as well as application of airdrop techniques to new operational situations.

2.2.1 Humanitarian Airdrop

A prime deterrent to the modern employment of aerial delivery in a high-threat environment has been the necessity for the delivering aircraft to fly at low altitude in order to achieve an accurate airdrop to the target and, thus, become vulnerable to ever more sophisticated anti-aircraft weaponry. Low-threat missions such as delivery of humanitarian supplies are not subject to the same risks and are still routinely executed. A recent dramatic example of a

humanitarian airdrop mission was as part of Operation UNIFIED RESPONSE, a relief effort to Haiti in the wake of a devastating earthquake in January 2010. One C-17 Globemaster III aircraft was able to deliver nearly 15 000 meals on one sortie; however, for operational reasons, the goods were delivered to a rural area instead of the place where the supplies were most needed, which was the capital city, Port-au-Prince [24].

Figure 2.3: Relief supplies delivered from the air to Haiti. Operation UNIFIED RESPONSE delivered supplies outside the capital Port-au-Prince in 2010. U.S. Air Force photograph.

2.2.2 Precision Airdrop

In the 1990s, various research teams, including one at NPS, began investigating the development of autonomous control systems that could guide an airdropped payload to a precise landing location [25], [26]. Since that time, this research area has blossomed. A recent overview presentation lists 14 modern ADSs, ranging in payload capacity from just 2 kg to over 13 500 kg. These modern systems share a common feature in that they are all designed to land on a fixed, immobile, target on the ground [27].

2.3 Beginnings of Maritime Use

Recently, there have been some demonstrations that illustrate the exciting possibilities of applying aerial delivery for maritime missions. Surprisingly, one of the most publicized demonstrations came not from naval forces but from the continuing piracy epidemic around the Horn of Africa. In January 2009, the British Broadcasting Corporation reported on a

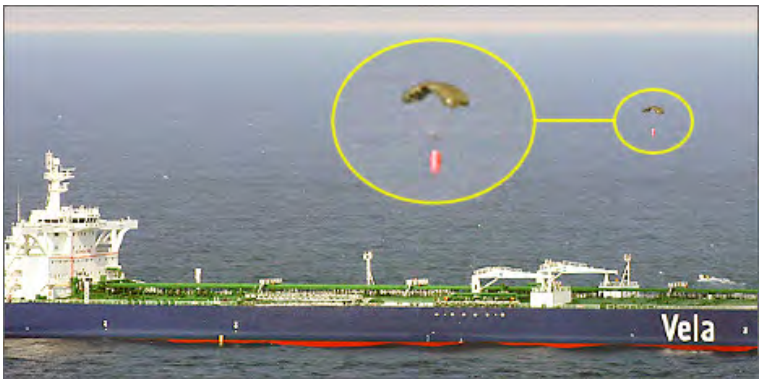


Figure 2.4: Aerial delivery of ransom payment to pirates. The detail in the upper-right is of a parafoil in flight during an apparent aerial delivery of payment to pirates holding the MV *Sirius Star*. U.S. Navy photograph with annotation by the British Broadcasting Corporation.

case where ransom payment was delivered to pirates who had seized the MV *Sirius Star* near Somalia [28]. An accompanying U.S. Navy photograph, shown in Figure 2.4, shows a container, presumably holding the ransom payment, flying toward the deck of the *Sirius Star* using a parafoil canopy. This example of ransom delivery using airdrop leads to the idea of delivering needed supplies to naval vessels at sea using precision aerial delivery.

Another demonstration of aerial delivery in a maritime setting was from a recent exercise in the Mediterranean Sea. In May 2007, during exercise FLEXIBLE LEADER 07, the Maritime Craft Air Delivery System (MCADS) was tested. During this exercise, two C-130 cargo aircraft delivered two Naval Special Warfare 11-meter Rigid Inflatable Boats into the operations area, with each drop using four cargo parachute canopies [29]. From this example, one can envision precision aerial delivery being used to place unmanned surface vehicles (USVs) or autonomous underwater vehicles (AUVs) at precise starting positions for their missions.

Again, it may be logistics that provides the compelling motivation for the application of precision aerial delivery in the maritime domain. Evidence is already emerging that the need for quick and flexible delivery to ships underway can be achieved using precision aerial delivery. In the spring of 2011, the USCGC *Bertholf* (WMSL-750), the first of a new class of U.S. Coast Guard National Security Cutters, began a patrol of the maritime boundary line in the Bering Sea. During her patrol, a critical part for the on-board helicopter was delivered using aerial delivery from a U.S. Coast Guard HC-130 aircraft [30]. The deliv-

Figure 2.5: Maritime Craft Air Delivery System (MCADS). A Naval Special Warfare 11-meter Rigid Inflatable Boat is delivered during exercise FLEXIBLE LEADER 07. U.S. Navy photograph.

ery of the canister containing the part is illustrated in Figure 2.6, where a round parachute was used and dropped from low altitude. The U.S. Coast Guard crew retrieved the floating package using a small boat, as illustrated in Figure 2.7. If precision aerial delivery had instead been used, the possibility exists that the part could have been delivered directly to the cutter's helicopter flight deck.

2.4 Potential for Maritime Use

The use of aerodynamic decelerators (e.g., parachutes and parafoils) and aerial delivery is not new to naval platforms; both sensors and weapons are already deployed from aircraft using this technique. For example, the U.S. Navy's P-3C Orion aircraft and SH-60B and MH-60R Seahawk helicopters deploy acoustic listening sensors called sonobuoys that are stabilized and slowed during their fall with a small drogue parachute. Also, air-deployed torpedoes from both patrol aircraft and helicopters use a drogue parachute to stabilize the torpedo prior to water entry. The sonobuoy is a sensor that the U.S. Navy has employed for

Figure 2.6: Aerial delivery of an aircraft part. A U.S. Coast Guard HC-130 is shown deploying an aerial delivery system containing a critical aircraft part to USCGC *Bertholf* (WMSL-750) on patrol in the Bering Sea in May 2011. U.S. Coast Guard photo by Petty Officer 3rd Class Charly Hengen.

anti-submarine warfare (ASW) for nearly fifty years. While the acoustic sensor has been upgraded with improving technology, the drogue parachute system of modern sonobuoys is very similar to those of the past. Accuracy in sonobuoy placement is achieved not through a precision aerial delivery system but rather through standard tactics that call for the deploying aircraft to drop the sonobuoys from a low altitude. As outlined in previous work by Hewgley and Yakimenko [17], a series of studies was conducted to study various methods of improving the accuracy of sonobuoy placement from higher altitudes; yet, the incorporation of a self-steering mechanism that is common to precision aerial delivery systems outlined by Tavan [27] has not yet been the focus of serious research. With these points considered, precision placement of sonobuoys represents another potential for maritime use of precision aerial delivery.

Sonobuoys are not the only sensors that could benefit from placement using precision aerial delivery. The strategic science and technology plan published by the Office of Naval Research (ONR) lists Maritime Domain Awareness as one of its thirteen focus areas [31]. One of the objectives of this focus area is "Homeland and port defense monitoring," which includes "new systems for target identification and tracking using fixed and deployable

Figure 2.7: Aerial delivery payload being retrieved by small boat. A special purpose craft crew from USCGC *Bertholf* (WMSL-750) in the Bering Sea prepare to retrieve a floating canister containing a critical helicopter part. U.S. Coast Guard photo by Petty Officer 3rd Class Charly Hengen.

cueing systems." One possible deployable cueing system might consist of a covert sensor landed on a ship of interest using precision aerial delivery.

The U.S. Navy has also funded investigations into means of deploying torpedoes accurately from high altitudes, as summarized in a previous paper by Hewgley and Yakimenko [17]. The drawback to using precision aerial delivery for torpedo deployment is that the speed with which the torpedo can be delivered to its target point on the sea surface is extremely important. A slow flight down to the surface under a guided parafoil after deployment, even if very accurately delivered, would degrade the torpedo's effectiveness. Therefore, weapon delivery using aerodynamic decelerators represents an area of limited potential for the application of precision aerial delivery to the maritime domain.

The use of precision aerial delivery as a technique for providing swift and flexible logistics delivery to ships underway is an untapped area of research. As noted in previous work by Hewgley and Yakimenko, the continual improvement of the demonstrated landing accuracy of modern aerial delivery systems at events such as the biannual Precision Airdrop Technology Conference and Demonstration (PATCAD) have made it possible to consider

using this technology as a way to deliver cargo to ships underway [17]. The inherent flexibility and responsiveness of aerial delivery has the potential to create a revolution in naval logistics.

2.5 Current State of the Art in Precision Aerial Delivery

The field of research into precision aerial delivery and, especially, autonomous aerial delivery systems is new. Much of the current development falls under a program of record led by the U.S. Army called the Joint Precision Airdrop System (JPADS), which is managed from the U.S. Army NSRDEC in Natick, Massachusetts. One current focus of effort at NSRDEC is a Joint Capability Technology Demonstration (JCTD) entitled Joint Medical Distance Support and Evacuation (JMDSE) [2], [32]. This demonstration seeks to illustrate the use of MLW systems, those in a payload weight range of 5 kg to 77 kg, to deliver critical medical supplies and equipment to injured soldiers on a battlefield using precision aerial delivery. This project is representative of current research, with much of the effort spent on repackaging the aerial delivery system to be deployable from different platforms; for example, UASs and ensuring the reliability of the aerodynamic decelerator system. By contrast, it seems that relatively little research has been conducted into advancements that are being pursued in current UAS research, such as advanced guidance algorithms and sensors, networking, and vision systems. Furthermore, whereas autonomous shipboard landing has been an area of research in UASs, there is very little in the literature about such research for parachute or parafoil-based ADSs. The following subsections constitute a survey of some relevant research in the ares of sensors and guidance algorithms, wind estimation, and networking that specifically applies to ADSs.

2.5.1 Sensors and Guidance Algorithms

The U.S. Army's JPADS work began as a technology demonstration program in 2003 and was formalized as a program of record in 2007 [1], [33], [34], and GN&C research has been a part of the development of this system throughout its history. The airborne guidance unit (AGU) component of JPADS which executes the system's GN&C functions at first relied only on GPS to provide measurements of the system's position in 3D space; however, some recent research at NSRDEC has focused on a sodar sensor for a more accurate measurement of the system's height above the ground [35]. Draper Laboratory in Cam-

bridge, Massachusetts, under contract to NSRDEC, develops the software that performs the GN&C functions for JPADS [36]–[38]. The guidance module of this software computes a trajectory in terms of a commanded turn rate ψ_{cmd} but limits this turn rate such that its derivative with respect to altitude $\mathrm{d}\psi/\mathrm{d}z$ is below a maximum threshold value [38]. The commanded turn rate is tracked using a proportional-integral-derivative (PID) inner-loop controller, with the actual turn rate measured by newly-incorporated inertial sensors in the AGU [37].

The Aerodynamic Decelerator Systems Center (ADSC) at NPS was founded in 2001 with a mandate to investigate advanced concepts, particularly in the field of guidance, navigation, and control pertaining to aerodynamic decelerators, including both round parachutes, and rectangular-planform parafoils (`http://www.nps.edu/Academics/Centers/ADSC/index.html`). An early project was the development of an autonomous control system that could guide an airdropped payload under a standard U.S. Army G-12 round cargo parachute to a precise landing location on the ground [25], [26]. Since 2008, Yakimenko from NPS and Slegers from UAH have continuously developed a small prototype aerial delivery system that flies under a rectangular parafoil canopy. This system, known as *Snowflake*, has become a major focus of research at the ADSC. An image of the Snowflake prototype ADS in flight during experimentation at Camp Roberts, California, in May 2011 is presented in Figure 2.8, and characteristics of the Snowflake ADS are presented in Table 2.1. The latest version of Snowflake is integrated with the Arcturus T-20 UAS to form a complete delivery system known as *Blizzard* [39].

Table 2.1: Snowflake system characteristics. These values are for a set of prototypes developed by NPS and UAH and flight tested between 2008 and 2011.

Characteristic	Value	Units
dry weight	1.9	kg
canopy span	1.4	m
canopy chord	0.6	m
descent rate	3.7	m/s
forward speed	7.2	m/s
glide ratio	2:1	
minimum turn radius	15.2	m

Sensors and guidance algorithms are two major topics of research for the Snowflake system.

Figure 2.8: Snowflake autonomous parafoil in flight. The small-scale prototype aerial delivery system is flying autonomously toward a preprogrammed target.

Snowflake's autopilot's advanced suite of sensors includes three-axis accelerometer, rate-gyroscope, and magnetometer as well as a GPS receiver and a barometric altimeter [11]. Snowflake's advanced guidance algorithm includes model predictive control for the path-following steering commands and an optimal control algorithm for terminal guidance trajectory planning [6], [7]. A history of Snowflake flight tests between 2009 and 2011 is presented in Appendix B. The majority of these tests were conducted to support the improvement of guidance to a fixed target on the ground; however, moving target experiments were conducted in February and May 2011.

In addition to the JPADS and Snowflake research, there have been several recent research efforts specifically investigating terminal guidance algorithms using simulation. Rademacher investigated the use of optimal control techniques to determine a final trajectory based in part on Dubins paths [40]. A Dubins path is a path constructed for a vehicle using the assumption that the vehicle can only move forward at a constant speed or turn with a turning rate $\dot{\psi}$ that is limited by a maximum turn rate $\dot{\psi}_{\max}$ such that $|\dot{\psi}| \leq \dot{\psi}_{\max}$. This motion is called *nonholonomic*, meaning that the vehicle cannot perform arbitrary translations in the two-dimensional plane. Parafoils using real differential brake steering tend to fly with a constant forward velocity and have the nonholonomic movement constraint in that they can either fly straight or turn up to a maximum turn rate. Because the Dubins path is associated with minimum time problems, and the parafoil terminal guidance problem is an exact

time problem, Rademacher proposes the idea of modified Dubins paths, in which a hybrid scheme is developed that uses a minimum-control, exact-time trajectory combined with minimum-time Dubins path segments for the final portions of the trajectory. Rademacher's technique, like that of Slegers and Yakimenko, plans for the final trajectory to terminate with a landing directly into the assumed wind.

Fowler and Rogers also propose a technique with which the final trajectory can be constructed with an upwind landing. They rely on Bézier curves instead of Dubins paths so that they can achieve flexibility in selecting a final trajectory to avoid known obstacles and also flexibility in selecting the flight duration of the final trajectory [41]. Rogers and Slegers address the problem of planning a final trajectory when the target is located among complex terrain features such as ridge lines or canyons [42]. This problem is very relevant to the current use of ADSs to resupply forward operating bases (FOBs) in the mountainous regions of Afghanistan. Rogers and Slegers propose a novel technique in that their algorithm treats the horizontal wind profile as a stochastic process. To deal with this, the algorithm executes many simulations simultaneously, each with a different perturbation to a mean wind profile, and each simulation producing a candidate trajectory. One of these many candidate trajectories is chosen based on predicted landing accuracy, accounting for the effect of the complex terrain on landing accuracy. For example, a trajectory for which a slight overshoot of the target leads to the ADS plunging down an adjacent canyon would be rejected in favor of a trajectory in which the final approach direction minimized that risk. The trajectory planning algorithms discussed in this subsection represent the current state of the art in this field; although all of them still assume a fixed target on land rather than a moving landing platform at sea. The work in this dissertation addresses the new challenge of the moving target.

2.5.2 Wind Models and Estimation

The axiom that gliding ADSs are at the mercy of the winds as they strive for accurate landings was introduced in Section 1.1.3. A consequence of this axiom is that, to plan for an accurate landing, the ADS's guidance algorithm needs some estimate of the wind velocity that will confront the ADS during its descent toward the target. One indicator of the importance of accurate wind knowledge to ADSs is the significant amount of work that has been done to gather accurate wind measurements.

The U.S. Army's Yuma Proving Ground (YPG) is a test range in Yuma, Arizona where much testing of ADSs occurs. Test engineers at YPG have over time refined their methods of measuring ambient winds. Kelly and Peña describe the original balloon-borne instrumentation, or radiosonde, used to measure horizontal wind velocity in comparison to the WindPack air-deployed probe, or dropsonde, developed at YPG [43]. Further study by Rogers compared data from these two systems with a U.S. Air Force consolidated wind prediction system [44], and Fraser documented improvements and further testing on the WindPack [45]. Most recently, Herrmann proposed the use of a ground-based lidar wind measurement system to transmit real-time information to an ADS in flight [46].

The idea of making real-time wind profile information available to an ADS in flight has potential to improve the accuracy of the planned final trajectory because the ADS itself can only make wind estimates at its current altitude. Several current ADSs still rely on the simple assumption that the wind speed and direction measured at the ADS's current altitude is also a valid representation of the wind profile for the rest of the way down to the surface. The AccuGlide ADS, developed by NSRDEC, uses a technique described by Bergeron as *glideslope surface guidance*, whereby the ADS uses the aforementioned constant wind profile assumption to compute a 3D surface along which to fly to the target [47], [48]. The AccuGlide uses features of its parafoil canopy that allow direct control of glideslope; this additional control can be used to counteract inaccuracy introduced by the constant wind profile assumption. The ParaLander system, developed by European Aeronautic Defence and Space Company (EADS) subsidiary Cassidian, uses an assumption of constant wind profile to construct a wind drift trajectory: the trajectory that would be flown by the ADS if no steering control were applied. Altmann describes how this system estimates horizontal wind velocity in flight; and, although it relies on a constant wind profile assumption at the beginning of its flight, the ParaLander assumes a linearly decreasing wind profile for its descent through the final 250 m to the surface [49], [50].

These wind profile assumptions are still too simplistic, claimed Ward in his examination of higher-fidelity wind models for ADS simulations [51]. He proposed conducting ADS simulations using more sophisticated stochastic wind models from the meteorology community so that system designers can obtain a more realistic assessment of what an ADS's accuracy might be. Calise makes a further argument that not only accuracy, but also dynamic

stability of an ADS in flight, may be degraded if wind knowledge is not accurate [52]. One additional recent paper by Yakimenko and Slegers [9] takes another look at accounting for the effect of one, two, and three-dimensional winds for Snowflake's final approach trajectory optimization algorithm. This paper considers constant, linear, and logarithmic horizontal wind profiles in the direction aligned with Snowflake's approach path as well as methods for adding crosswind and vertical wind components to the calculations. The logarithmic wind profile was first presented by Hewgley and Yakimenko in 2011 [18] and is further explored in this dissertation as a model that aims for a middle ground between models that are simple but unrealistic and those that are realistic but overly complex.

The current ADS research described in the literature, as listed in this section, has so far failed to address the problem of landing on a moving target in a maritime environment. On the other hand, the experiment of landing a parafoil vehicle on a ship has, in fact, been done. In a 1991 paper entitled *Parafoils for Shipboard Recovery of UAVs*, G. Brown describes experiments, including at-sea flight tests, of a system for landing a fixed-wing UAS [53]. Using this system, the UAS would deploy a parafoil canopy from the top of its fuselage to achieve a slow, controlled descent to the deck. The key differences between this experiment and the problem at hand is that, in Brown's experiment, the UAS kept engine power on, so that the operator could use the vehicle's throttle to control its rate of descent. Also, this system relied on a human operator instead of an automatic control algorithm. Nevertheless, Brown's paper does contain good observations on the mechanics of landing a parafoil-borne object on to a moving deck at sea. The next section contains a survey of research in other disciplines that is relevant to certain aspects of the shipboard landing problem, particularly to target estimation and to wind profile modeling.

2.6 Relevant Research in Other Fields

The previous section contained a survey of current ADS research which focused on autonomous ADSs seeking to land accurately on a fixed target on land. For the single instance of parafoil shipboard landing research found, Brown's experiment did involve a parafoil and a landing vehicle; however, it was not an autonomous parafoil—a human operator landed the vehicle using remote control. A logical topic with which to start the survey of other relevant work is that of autonomous shipboard landing. First, though, it is necessary to survey the fields of computer vision and image processing because visual sensing plays a

major role in several of the GN&C schemes of the aerial vehicles that will be surveyed. In particular, the fields of image and video processing and computer vision both have methods for deducing information about a three-dimensional scene from its two-dimensional image.

2.6.1 Computer Vision and Image Processing

In the last three decades, a very rich body of literature has been developed within the topics of image and video processing and computer vision. A dividing line within this body of literature was stated elegantly by Woods [54]: "image-in/image-out" is a succinct description of problems that can be categorized within image and video processing, whereas "image-in/analysis-out" describes problems that are traditionally grouped under the title computer vision. Another author, Umbaugh, also addressed the point expertly by noting that image processing concerns images and data for human consumption, whereas computer vision involves processing images for use by a machine [55]. Because the aerial delivery systems addressed in this dissertation are autonomous machines, designed to act without human intervention in flight, the use of visual sensor data in this context fits the definition of an application of computer vision. In their classic text on computer vision, the authors Ballard and Brown state that computer vision "is the construction of explicit, meaningful descriptions of physical objects from images" [56]. An explicit, meaningful description, consisting of position and velocity, is precisely what the ADS should derive from a sequence of images of the target.

A subtopic common to both image processing and computer vision is that of motion estimation. Within the realm of image processing, motion estimation is done as a precursor to such tasks as video compression and video sampling rate conversion. For computer vision, the goal is to deduce structure and motion of an object in three dimensions, using information from two-dimensional images, as explained by Yang [57]. Specifically, in the case of an ADS attempting to land on a moving platform that is not transmitting its position, the ADS must compute the vectors representing the position and velocity of the target relative to the ADS using information from the visual sensor together with other on-board sensors. In a seminal paper on the topic of motion estimation, Aggarwal and Nandhakumar identified two major families of motion estimation techniques: optical-flow-based (also called intensity-based) and feature-based [58]. The intensity-based methods are designed to compute a two-dimensional field of velocities of pixels in the image plane by considering

changes in pixel intensity values over time. The intensity-based methods are considered to be simpler than feature-based methods. Using the latter, one must first analyze each image in order to locate a set of features which can then be tracked frame-to-frame in order to compute motion in the two-dimensional image plane.

In the robotics community, using information from a visual sensor as part of a feedback loop is known as visual servo control, and intensity-based and feature-based motion estimation methods are sometimes expressed as *image-based* and *position-based* algorithms, respectively. A classic tutorial by Hutchinson [59] explores both of these methods with respect to fixed-base robotic arms with manipulators. In this dissertation, the terms *intensity-based* and *feature-based* will be used to described the two families of motion estimation techniques. As will be shown, a feature-based approach will be followed in addressing the problem of enabling an airborne ADS to estimate motion of a moving target landing platform. Note that the algorithms in this section assume that the image features have already been identified. The actual mechanics of feature extraction are beyond the scope of this work. The topic of motion estimation is particularly applicable to the problem of autonomous shipboard landing; therefore, this topic receives in-depth treatment in Section 2.6.3. First, a more broad range of methods for addressing the autonomous shipboard landing problem will be examined.

2.6.2 Autonomous Shipboard Landing

In examining autonomous shipboard landing examples, the most prevalent case in U.S. Naval operations is that of a rotary-winged aircraft, or *rotorcraft*, landing on small-deck air-capable ships.[2] The most relevant research to review pertinent to ADSs would be that involving UASs rather than rotorcraft with human pilots.

Rotary-wing UAS

Fortunately, an excellent and very thorough survey paper by Kendoul [60] has been published that catalogs and summarizes 27 research groups worldwide (excluding military and industrial groups) pursuing research and development of autonomous rotorcraft UAS.

[2] Active ships in service in 2013 include 19 large deck ships: 10 aircraft carriers (CVN) and 9 amphibious assault ships (LHA, LHD), and 101 small deck ships: 22 cruisers, 62 destroyers, and 17 frigates (CG, DDG, FFG). Source: `http://www.navy.mil/navydata/fact.asp`.

The key finding is that autonomous rotorcraft UAS shipboard landings have been demonstrated, not just by the U.S. Navy with the MQ-8B Fire Scout [61], but also by industry groups [62], and also academic research groups [63]. Unfortunately, these results for rotorcraft UAS may have little to offer for ADS experimental development because the most common rotorcraft UAS landing technique is the low overhead hover and straight vertical descent. This maneuver, being executed by the Fire Scout UAS as shown in Figure 2.9, is one that an ADS cannot perform.

Figure 2.9: RQ-8A Fire Scout UAS prepares for autonomous landing. The vehicle hovers above the deck before landing autonomously aboard the amphibious transport dock ship USS *Nashville* (LPD-13). U.S. Navy photograph, 17 January 2006.

Fixed-wing UAS

Research on autonomous shipboard landing of fixed-wing UAS, while much less common that that for rotorcraft, may prove more relevant. A fixed-wind aircraft would most likely fly a straight-in approach to the landing area, much as an ADS would do. In fact, shipboard landing of a fixed-wing UAS, including tests with flight hardware, has been a particular area of focus at NPS, with extensive hands-on work by Lizarraga [64], Kaminer [65], and Yakimenko [66]. Some of these efforts have relied on a differential GPS beacon on an ar-

resting net on the target vessel. This recovery gear setup would most likely not be available in the ADS landing scenario.

Distinctive visual markings and features and their use with a visual sensing system are a focus of fixed-wing shipboard landing research at the French Institut National de Recherche en Informatique et en Automatique (INRIA). Research published by Coutard [67], [68] considers certain features of an aircraft carrier landing deck that the landing algorithm uses to compute the relative pose between the landing aircraft and the deck. The landing algorithm uses this relative pose to control the aircraft along a constant flightpath descent to the deck. This work's emphasis on visual estimation of relative pose is relevant to the problem at hand of landing an ADS, even though the French research focused on human-piloted fighter aircraft landing on a full-sized aircraft carrier.

Another group at Brigham Young University (BYU) has actually demonstrated autonomous landing of a fixed-wing UAS first to a fixed point on the ground [69], then to an actual moving platform [70]. In the latter case, the platform was not a ship but a pickup truck driving on a runway. The earlier BYU research by Barber [69] focused on estimating the height above ground level (AGL) of the aerial vehicle by combining measurements from a visual sensor designed for optic flow measurements with velocity measurements from GPS and barometric altimeter measurements. Accurate estimates of vehicle height AGL were shown to enable accurate landings to a fixed point on the ground with *known* coordinates. Barber's later BYU research [70] relaxed the requirement for a precisely known target location and instead used a vision-based glidepath tracking algorithm once the target was in the visual sensor's field of view. The vision-based control algorithm was designed to handle constant-velocity target movement by treating this motion as an extra component of estimated wind. The visual guidance algorithm described in this work had the advantage of requiring no modifications to the target vehicle save distinctive markings in the landing area (bed of the pickup truck).

One additional autonomous landing research effort conducted at the Georgia Institute of Technology (GT) is notable for its incorporation of optimal estimation techniques. Proctor and Johnson in 2004 [71] equipped a small glider with only a vision sensor and designed an extended Kalman filter (EKF) that allowed the glider to estimate its relative pose with respect to the landing target, which was an open window on one wall of a small rectan-

gular structure. These two researchers furthered their research in 2005 [72] when they equipped the GTMax autonomous helicopter with an updated visual sensing algorithm that used an EKF to estimate the aircraft's pose with respect to colored markers at the corners of a landing runway. In this case, the GTMax was programmed to fly a standard fixed-wing approach to the runway instead of a helicopter hover-to-land approach to evaluate the suitability of this algorithm to a fixed-wing UAS.

It is interesting to note that both the BYU and GT vision-based UAS navigation research efforts evolved to include motion estimation of moving targets: ground targets for BYU and airborne targets for GT. These motion estimation research efforts along with others will be explored in the next section after a more general overview of motion estimation is presented.

2.6.3 Motion Estimation

In considering the overarching problem of estimating motion of some target based on limited measurements available to an observer, one may realize that this is a problem that the U.S. Navy, the submarine community in particular, has addressed under the label target motion analysis (TMA) [73]. TMA is used in situations in which a submarine attempts to determine course and speed of a target, which could be an adversary submarine or surface ship, using passive (*bearings-only*) sonar. Subsequent research in mobile robotics has extended this problem definition to include the estimation of several combinations of observer and target states using passive or active (ranging) sensors. Frew [74] presents a clear and logical categorization of the various estimation scenarios as *target-motion estimation* (observer state known, target state estimated), *localization* or *navigation* (target state known, observer state estimated), and *simultaneous localization and mapping* (SLAM) (partial knowledge of, and estimation of, both observer and target states). The problem of an airborne ADS seeking a moving landing platform on the surface is firmly in the category of target motion estimation.

Monocular Vision, Stereo Vision, and Range-finding Sensors

Before proceeding further, the question should be addressed of why monocular vision (one camera) is used and not binocular, or stereo, vision (two cameras). For the application of an ADS attempting to land on a moving target on the ocean's surface, it was presumed

that cheaper and more expendable sensors would be preferable to more complex and expensive systems. Binocular, or stereo vision systems, even though declining in price, still require baseline calibration between the two visual sensors and more sophisticated video processing than do monocular systems.

The same argument of too much complexity can be made against active, or range-finding sensors such as radars, laser rangefinders, and lidars. Compared to these alternative sensors, a monocular visual sensor would likely be lighter, cheaper, and consume less power. For these reasons, it was decided that for the purposes of this research, only methods that could be applied to a monocular system would be considered.

Airborne Observer and Moving Target

The BYU team achieved the moving target landing described in Section 2.6.2 by estimating the position and velocity of the target relative to the observer [70]. Barber demonstrated geolocation of a fixed ground target by constructing a circular orbit around the target and performing RLS estimation on the raw visual measurements of target position [75]; geolocation of a moving ground target would involve a combination of these two techniques. Subsequent work by Johansen [76] focused on stabilization of video from the airborne visual sensor and synchronization of video with other flight data.

The GT target motion estimation technique was developed in the context of air-to-air pursuit, formation flying, or docking (such as for aerial refueling). Johnson developed an EKF for estimating both the position and velocity of the airborne target relative to the airborne observer [77]. The challenge when both observer and target are airborne is computing the distance between the vehicles in three dimensions using an image plane measurement in two dimensions. Johnson details two techniques to overcome this challenge: the first is to maneuver the observer laterally and estimate the range using multiple views; the second is to estimate range using a known size of the target (the target's wingspan).

The problem of tracking a ground moving target by an airborne fixed-wing observer was addressed in recent work by T. Oliveira at the Portuguese Air Force Academy [78]. In this paper, trajectory planning was the focus: a fixed-wing observer computed a fixed-radius orbit around a moving ground target so that no matter the target's motion, the airborne observer could maintain a fixed-radius orbit around the target. Motion estimation was not

addressed in this paper; the target's position was assumed to be known, not estimated.

At NPS, both tasks of trajectory planning and target motion estimation were investigated in a series of experiments. A solution to the problem of estimating the range from an airborne vehicle to an object on the earth's surface was proposed by Kaminer, Kang, Yakimenko, and Pascoal for the application of automated landing of a fixed-wing UAS on to an underway vessel [79]. In this work, the geometry and kinematics of the flying vehicle and surface landing platform are exploited in order to estimate the range (or the position of the landing platform relative to the flying vehicle) based on the flying vehicle's own measured angular orientation, its height above the surface, and the two-dimensional coordinates of the image of the landing platform as measured by the visual sensor.

The method used is a nonlinear estimator that combines the visual sensor measurements with those of an inertial measurement unit (IMU) on-board the aerial vehicle. In addition to the original work, this same estimation method has also been applied to the estimation of the relative position and velocity of an AUV with respect to a USV for the application of oceanographic research. In a dissertation by P. Oliveira [80], this estimation algorithm along with visual sensing of a strobe light attached to the submerged AUV is proposed as the method by which the USV can track and maintain station above the AUV so that a vertical acoustic communication link can be used between the two vehicles. The original application of tracking of a moving target on the surface by an aircraft equipped with a visual sensor was extended by Hespanha [81], who extended the method to handle the case in which the target was located outside the field of regard of the visual sensor for short periods of time. Finally, this nonlinear estimation was again used by Dobrokhodov [82] for a UAS with a visual sensor mounted in a turret. In this case, the UAS was tasked with calculating a trajectory such that a ground moving target could always be kept centered in the turret-mounted camera's field of regard.

2.7 Relationship to Research in this Dissertation

The contributions of this dissertation detailed in Section 1.3 are similar to, yet go beyond, the related research surveyed in this chapter. The target estimation algorithm developed in Chapter 3 has some aspects in common with the air-to-air EKF estimation algorithm of Johnson [77] and the air-to-ground nonlinear estimation algorithm of Dobrokhodov [82].

The estimation algorithm in Chapter 3 relies on the Unscented Transform to characterize nonlinear measurement errors, whereas the other two works use Jacobian matrices. Also, the algorithm in Chapter 3 uses the epipolar constraint with the last-in-view and first-in-view measurement samples; therefore, it is able to tolerate long periods of time during which the target is out of view.

The wind profile modeling algorithm described in Chapter 4 builds on the previous work of Hewgley and Yakimenko [18] concerning the logarithmic wind model. The algorithm described in Chapter 4 will determine when to activate an optimal final turn algorithm such as that described by Yakimenko and Slegers [9]; the work in this dissertation does not replace the optimal final turn algorithm. The work in this dissertation does improve on the constant and simple linear horizontal wind profiles used in the AccuGlide [47] and ParaLander [49].

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 3:
Visual Sensing for Terminal Guidance

This chapter contains the method of estimating the target's position and speed using measurements of a two-dimensional image. The explanation of this method begins with a review of the geometry of projection, and the homogeneous coordinate system that underpins the calculations. Next, a state-space formulation is developed for estimating the target's motion. The model is first developed using a simple set of state variables. Later, a more realistic set of state variables is substituted, and a method for dealing with the out-of-frame condition is introduced. The chapter concludes with results of a simulation designed to measure estimator performance.

3.1 Projecting a 3D Scene on a 2D Image Plane

The ease with which the human brain grasps three-dimensional structure from the two-dimensional images provided by the human visual system belies the difficulty of the 3D to 2D transformation. A machine that attempts this same task needs a working mathematical relationship between each object point location in the three-dimensional scene, and the corresponding image point location on the two-dimensional image plane. This relationship, known as the *perspective-projective transformation*, is presented clearly in the text by Schalkoff [83]. The perspective-projective transformation, then, is a mapping of object coordinates $\mathbf{x}_o \in \mathbb{R}^3$ to image coordinates $\mathbf{x}_i \in \mathbb{R}^2$. Critical to this transformation is the selection of a coordinate system.

3.1.1 Using Homogeneous Coordinates

An effective technique for using matrix multiplication to calculate this transformation is the use of *homogeneous coordinates*, also explained by Schalkoff in his thorough introduction to this topic [83, ch. 2]. To convert object coordinates $\mathbf{x}_o \in \mathbb{R}^3$ and image coordinates $\mathbf{x}_i \in \mathbb{R}^2$ to homogeneous coordinates, multiply the coordinate vectors by scale factors w_o and w_i, respectively, and also append these scale factors as an additional element to each of these vectors. An overbar denotes the homogeneous versions of the coordinate vectors, as $\bar{\mathbf{x}}_o \in \mathbb{R}^4$ and $\bar{\mathbf{x}}_i \in \mathbb{R}^3$.

Thus, let $\bar{\mathbf{x}}_o$ be the four-dimensional vector describing the coordinates of an object in the 3D world, and let $\bar{\mathbf{x}}_i$ be the three-dimensional vector describing the coordinates of an image of that object on the two-dimensional image plane:

$$\bar{\mathbf{x}}_i = \begin{bmatrix} w_i u \\ w_i v \\ w_i \end{bmatrix} \qquad \bar{\mathbf{x}}_o = \begin{bmatrix} w_o x \\ w_o y \\ w_o z \\ w_o \end{bmatrix}, \tag{3.1}$$

where (u, v) are the coordinates of the image of the object on the 2D image plane.

Assuming that the coordinates of both image and object are defined with respect to the same coordinate axes, and using homogeneous coordinates, the perspective-projective transformation can then be represented by matrix $\mathbf{P} \in \mathbb{R}^{3\times 4}$, such that

$$\bar{\mathbf{x}}_i = \mathbf{P}\bar{\mathbf{x}}_o. \tag{3.2}$$

Matrix $\mathbf{P}$ from Equation (3.2) is known as the *projection matrix*, and it is the next topic.

3.1.2 Perspective-Projective Transformation

Consider a coordinate frame with origin O_i located in the image plane, and with mutually orthogonal axes X_i, Y_i, and Z_i as shown in Figure 3.1. The subscript "i" emphasizes the fact that this coordinate frame is anchored in the image frame. This is the front-projection pinhole-camera model described by Schalkoff [83], whose origin is the center of the image plane, and whose X_i axis is pointed down the optical axis of the sensor. The focal point in Figure 3.1 (the actual pinhole in a real pinhole camera) is located *behind* the image plane (opposite from the direction of the object) at coordinates $(-f, 0, 0)$. In the front-projection model, the image plane is located between the focal point and the object; therefore, the image of the object is not inverted as it is in a real pinhole camera. The coordinates of the object in this frame are (x_o, y_o, z_o). The coordinates of the object's image on the image plane are as follows: coordinate u is the measure of the image position along the image plane Y_i axis, and coordinate v is the measure of the image position along the image plane Z_i axis.

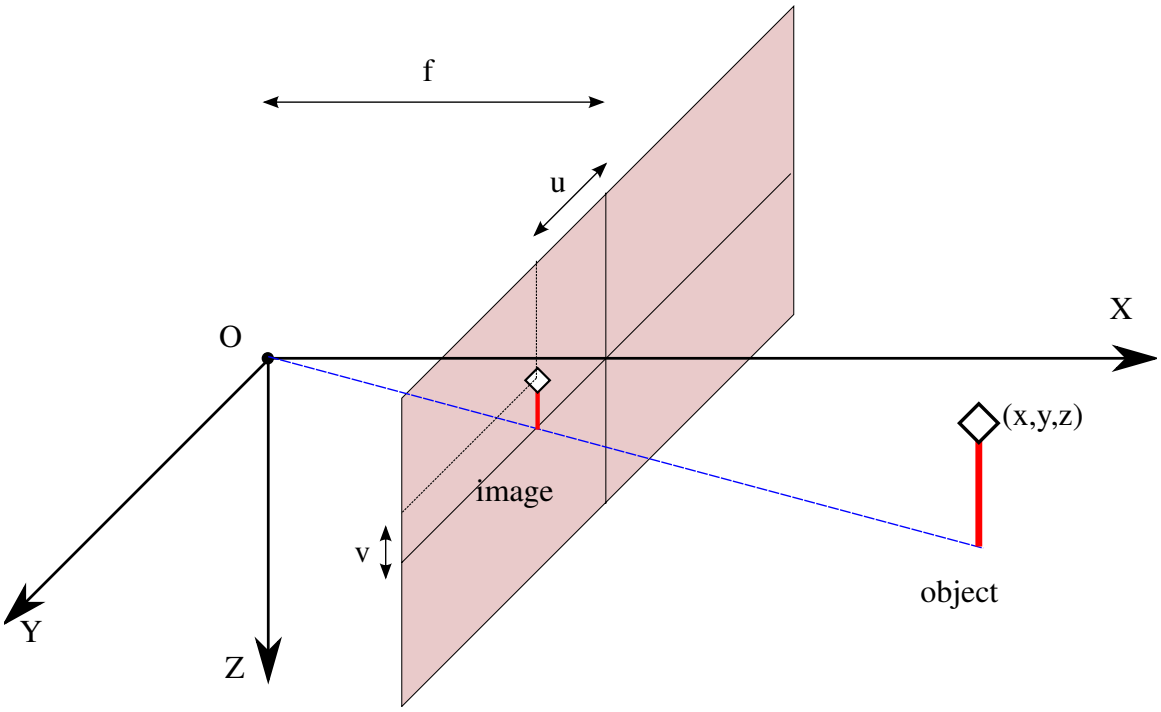


Figure 3.1: Geometry of the perspective-projective transformation. The object with coordinates (x,y,z) in the 3D world is projected on to a 2D image plane. The coordinates of the image are (u,v).

The geometry of similar triangles leads to expressions for the image plane coordinates (u,v) in terms of the object coordinates (x_o,y_o,z_o):

$$u = \frac{fy_o}{x_o+f} \qquad v = \frac{fz_o}{x_o+f}. \tag{3.3}$$

In homogeneous coordinates, these relationships can be written in matrix form:

$$\begin{bmatrix} w_i u \\ w_i v \\ w_i \end{bmatrix} = \begin{bmatrix} 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 1 & 0 & 0 & f \end{bmatrix} \begin{bmatrix} w_o x_o \\ w_o y_o \\ w_o z_o \\ w_o \end{bmatrix}. \tag{3.4}$$

The fourth row of this matrix equation sets the ratio between scale factors w_o and w_i:

$$w_i = w_o(x_o+f) \Rightarrow \frac{w_o}{w_i} = \frac{1}{x_o+f}. \tag{3.5}$$

The assumption that $x_o \gg f$ is known as the *far-field* assumption; applied to Equation (3.3), it leads to the following approximations:

$$u \approx \frac{f y_o}{x_o} \qquad v \approx \frac{f z_o}{x_o}. \tag{3.6}$$

The approximation in Equation (3.6) allows the lower-right element of the projection matrix in Equation (3.4) to be set to zero, resulting in the revised projection relationship:

$$\begin{bmatrix} w_i u \\ w_i v \\ w_i \end{bmatrix} = \begin{bmatrix} 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w_o x_o \\ w_o y_o \\ w_o z_o \\ w_o \end{bmatrix}. \tag{3.7}$$

Then, the first and second rows of Equation (3.7) yield equations that mimic the approximations of Equation (3.6). Thus, Equation (3.7) represents the same equation as (3.2) but written in expanded form. In practice, both sides of (3.7) may be divided by a scale factor w_i, so that the homogeneous coordinates for the image are normalized to unity, and the scale factor for the homogeneous coordinates of the object becomes $w' = w_o/w_i$, yielding:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} w' x_o \\ w' y_o \\ w' z_o \\ w' \end{bmatrix}. \tag{3.8}$$

3.1.3 Frames of Reference

Next come the specifics of how the coordinates for the 3D scene and the 2D image are defined; in general, the *global* coordinate frame used to describe objects in the 3D scene will have a different origin and orientation from the *image* coordinate frame that is aligned with the image plane. The goal of this section is to describe the transformation between one set of coordinates and another.

Both global and image coordinate frames are shown together in Figure 3.2. The global coordinate frame has axes x_{g}, y_{g}, and z_{g}, and its origin is O_{g} at the lower left of Figure 3.2.

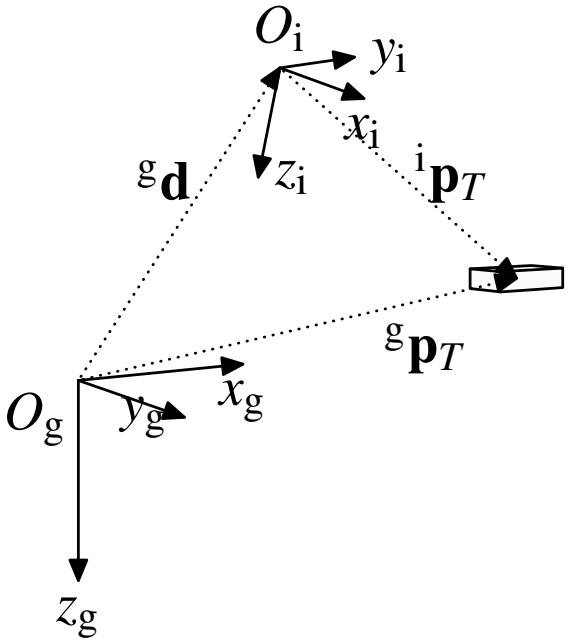


Figure 3.2: Overview of the geometry of the global and image reference frames. The locations of the observer and the target are each measured with respect to a global reference coordinate frame. An image reference frame is associated with the observer.

Two vectors are defined with respect to this coordinate frame: ${}^{\mathrm{g}}\mathbf{p}_T$ is the position vector of the target, and ${}^{\mathrm{g}}\mathbf{d}$ is the displacement, or position, vector of the observer. The leading superscript g denotes that both of these vectors are defined with respect to the global coordinate frame. The image coordinate frame has axes x_{i}, y_{i}, and z_{i} and it has its origin O_{i} at the head of vector ${}^{\mathrm{g}}\mathbf{d}$. The arrangement of the three axes of the image frame $\{i\}$ is the same as shown in Figure 3.1, and follows the convention set by earlier work on vision-based navigation for UASs conducted at NPS [84], [85]. One vector ${}^{\mathrm{i}}\mathbf{p}_T$ is defined in the image frame $\{i\}$: this is the position in 3D of the target relative to the observer; this is different from the 2D coordinates (u, v) of the image in the image plane.

Next, let the set of three angles, (ϕ, θ, ψ), called Euler angles, represent a rotation from the global coordinate frame $\{g\}$ to the image coordinate frame $\{i\}$. Let the overall rotation be composed of a set of three sequential elementary rotations; therefore, the order in which the three elementary rotations are performed is significant. The conventional sequence is: first, rotate by angle ψ around Z_g, then by angle θ around Y_g, finally by angle ϕ around X_g. This sequence mirrors the standard convention for rotating from a global frame $\{g\}$

to an aircraft body-fixed frame $\{b\}$ used in aircraft flight dynamics as described in the standard reference by Schmidt [86]. In this case, these elementary rotations transform a set of coordinates from $\{g\}$ to $\{i\}$.

For this problem, let $\mathbf{\Lambda}$ represent the triplet of Euler angles (ϕ, θ, ψ); therefore, the rotation matrix needed to rotate frame $\{g\}$ to align with frame $\{i\}$ is a function of $\mathbf{\Lambda}$. The reverse relationship is also needed: a matrix that rotates frame $\{i\}$ to align with frame $\{g\}$, which is simply the inverse of the matrix described in the previous sentence. Between two orthogonal coordinate systems so defined, such rotation matrices are orthonormal, such that

$$\mathbf{R}^{-1} = \mathbf{R}^{\mathrm{T}}. \tag{3.9}$$

Therefore, let $\mathbf{R}(\mathbf{\Lambda})$ represent a rotation matrix that rotates $\{i\}$ to align with frame $\{g\}$. The vector relationship shown in Figure 3.2 can now be expressed mathematically as:

$${}^{\mathrm{g}}\mathbf{p}_T = {}^{\mathrm{g}}\mathbf{d} + \mathbf{R}(\mathbf{\Lambda}){}^{\mathrm{i}}\mathbf{p}_T. \tag{3.10}$$

The two reference frames so described, the global reference frame $\{g\}$ and the image reference frame $\{i\}$, have different origins and different orientations. Equation (3.10) indicates that both rotation and translation (vector addition) are required to convert the coordinates of the target with respect to the observer, ${}^{\mathrm{i}}\mathbf{p}_T$ to position coordinates with respect to the global frame, ${}^{\mathrm{g}}\mathbf{p}_T$. A position vector such as ${}^{\mathrm{i}}\mathbf{p}_T$ is therefore known as a *bound vector*, since its magnitude, direction, and its end points must be known for the vector to have meaning. The use of homogeneous coordinates, as introduced in Section 3.1.1, allows the necessary rotation and translation for a bound vector to be completed as one matrix multiplication.

It can be shown (for example in Schalkoff [83]) that a single affine transformation $\mathbf{T}$ consisting of a rotation and a translation can be accomplished on vector $\bar{\mathbf{x}}$ (recall the overbar denotes homogeneous coordinates) using the following relationship:

$$\bar{\mathbf{x}}' = \underbrace{\begin{bmatrix} \mathbf{R} & \mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix}}_{\mathbf{T}} \bar{\mathbf{x}} \tag{3.11}$$

where $\bar{\mathbf{x}}'$ denotes the transformed vector in homogeneous coordinates, $\mathbf{R} \in \mathbb{R}^{3\times3}$ is a rotation matrix, and $\mathbf{d} \in \mathbb{R}^{3\times1}$ is the displacement (translation) vector.

Therefore, the transformation needed both to rotate $\{i\}$ to align with $\{g\}$ and also to translate the origin of $\{i\}$ to coincide with the origin of $\{g\}$ would be:

$$\mathbf{T}(\mathbf{\Lambda}, {}^{\mathrm{g}}\mathbf{d}) = \begin{bmatrix} \mathbf{R}(\mathbf{\Lambda}) & {}^{\mathrm{g}}\mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.12}$$

where the vector ${}^{\mathrm{g}}\mathbf{d}$ once again is the displacement vector of the observer from the origin of $\{g\}$ expressed in global coordinates.

Equation (3.12) provides the matrix needed to multiply a bound vector with homogeneous coordinates in $\{i\}$ to transform it to a bound vector in $\{g\}$. The inverse relationship is also required: transformation from $\{g\}$ to $\{i\}$. This inverse relationship is:

$$\mathbf{T}^{-1}(\mathbf{\Lambda}, {}^{\mathrm{g}}\mathbf{d}) = \begin{bmatrix} \mathbf{R}^{\mathrm{T}}(\mathbf{\Lambda}) & -\mathbf{R}^{\mathrm{T}}(\mathbf{\Lambda}){}^{\mathrm{g}}\mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix} \tag{3.13}$$

which can be verified by computing $\mathbf{T}(\mathbf{\Lambda}, {}^{\mathrm{g}}\mathbf{d})\mathbf{T}^{-1}(\mathbf{\Lambda}, {}^{\mathrm{g}}\mathbf{d}) = \mathbf{I}$.

Thus, the result of employing the the transformation in Equation (3.13) is:

$$\begin{bmatrix} {}^{\mathrm{i}}x_T \\ {}^{\mathrm{i}}y_T \\ {}^{\mathrm{i}}z_T \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{R}^{\mathrm{T}}(\mathbf{\Lambda}) & -\mathbf{R}^{\mathrm{T}}(\mathbf{\Lambda}){}^{\mathrm{g}}\mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix} \begin{bmatrix} {}^{\mathrm{g}}x_T \\ {}^{\mathrm{g}}y_T \\ {}^{\mathrm{g}}z_T \\ 1 \end{bmatrix}. \tag{3.14}$$

The vector on the left-hand side is expressed in homogeneous coordinates in the image plane reference frame, but it expresses the coordinates of the *target itself* and not of the target's image on the image plane. Therefore, the vector on the left-hand side of Equation (3.14) is analogous to the vector on the right-hand side of Equation (3.4), although expressed without a scale factor because the projection operation has not been applied to Equation (3.14). The image coordinates (u, v) will be the result of a *projection* of coordinates $({}^{\mathrm{i}}x_T, {}^{\mathrm{i}}y_T, {}^{\mathrm{i}}z_T)$ on to the image plane.

3.1.4 Transformation of an Object's Global Coordinates

Now, in preparation for combining the transformation matrix developed in Section 3.1.3 with the projection matrix developed here, use of the labels *image* and *object* will be discontinued in favor of the labels *observer* and *target*.

In the next expression, the coordinates of the *image* of the target on the image plane are labeled u and v, where $u = {}^{\mathrm{i}}y$ and $v = {}^{\mathrm{i}}z$. The full transformation takes the vector of the 3D coordinates of the target on the world's surface as input, then rotates, translates, and projects this vector resulting in a vector of the 2D coordinates of the image of the target in the image plane:

$$\underbrace{\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}}_{\text{img. coords}} = \underbrace{\begin{bmatrix} 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix}}_{\text{projection}} \underbrace{\begin{bmatrix} \mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda}) & -\mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda})^{\mathrm{g}}\mathbf{d} \\ \mathbf{0} & 1 \end{bmatrix}}_{\text{rotation and translation}} \underbrace{\begin{bmatrix} w'^{\mathrm{g}}x_T \\ w'^{\mathrm{g}}y_T \\ w'^{\mathrm{g}}z_T \\ w' \end{bmatrix}}_{\text{target position}} . \tag{3.15}$$

The expression in Equation (3.15) is almost, but not quite, the one that is needed for the measurement equation of the estimation algorithm developed in Section 3.2. That measurement equation of the estimation algorithm takes as input measurements of the image location in the image plane, and produces as output the coordinates of the target in the global coordinate frame. In other words, the inverse of Equation (3.15) is needed; however, the dimensions of the matrix multiplication on the right-hand side result in a matrix that is not square, and therefore not invertible.

The geometry of the overall problem can be invoked once again to realize that the z coordinate of the target *must* be zero if the following assumptions are made: first, that the target is a ship on the sea; second, that the origin of the global coordinate system is at sea level; and third, that the effects of vertical wave motion and the height of the ship's landing area above the sea surface are neglected. The assumption that the target remains at zero height (${}^{\mathrm{g}}z_T = 0$) allows the third column of the rotation and translation matrix to be eliminated, along with the third element of the target coordinate vector.

Let the product of the 3×4 projection matrix and the 4×3 (reduced column) rotation and translation matrix be called $\mathbf{M}$. Then, the global coordinates of the target can be computed from the image plane measurements by:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \underbrace{\begin{bmatrix} 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \overbrace{\begin{bmatrix} \mathbf{R}^{\mathrm{T}}(\mathbf{\Lambda}) & -\mathbf{R}^{\mathrm{T}}(\mathbf{\Lambda})^{\mathrm{g}}\mathbf{d}_o \\ 0 \quad 0 & 1 \end{bmatrix}}^{3^{\mathrm{rd}} \text{ column removed}}}_{\mathbf{M}} \begin{bmatrix} w'^{\mathrm{g}}x_T \\ w'^{\mathrm{g}}y_T \\ w' \end{bmatrix} \tag{3.16}$$

$$\begin{bmatrix} w'^{\mathrm{g}}x_T \\ w'^{\mathrm{g}}y_T \\ w' \end{bmatrix} = \mathbf{M}^{-1} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix}. \tag{3.17}$$

The scale factor can be subsequently removed in the 3×1 vector on the left-hand side of Equation (3.17) by dividing the first two elements of the vector by the third element. It is necessary to have the scale factor on the left-hand side of Equation (3.17) because the scale factor is not known *a priori*, so it cannot be inserted into the image plane coordinate vector.

Equation (3.17) indicates a noteworthy result: because the target is constrained to a 2D plane, a *reverse projection* is possible. Given coordinates of the target's image on the image plane, the target's physical location on a 2D world surface can be computed. Such reverse projections are not in general possible because the perspective-projective transformation is not in general reversible. For example, given a photograph of a statue in a plaza, an observer cannot, without additional information, determine whether the statue is 2 m tall and located 10 m from the camera, or whether the statue is 4 m tall and located 20 m from the camera. In this case, the 2D-to-2D mapping of world surface to image plane is reversible, considering the constraints on the target's position. Note that the invertibility of matrix $\mathbf{M}$ is not assured; however, in the typical geometry of an observer looking down at a significant angle to the target, matrix $\mathbf{M}$ will likely not be ill-conditioned.

For the remainder of this chapter, let us assume that the image coordinates u and v are readily available. In other words, let us assume that the computer vision tasks of target

detection and segmentation from the background have been completed without error. Using this image information for estimating the state of the target is the focus of the subsequent sections.

3.2 Method for Target Motion Estimation

Now that the stage has been set, the motion estimation problem can be addressed directly. From this point on, the term *observer* will refer to the airborne visual sensor and the term *target* will refer to the landing platform, which is assumed to be moving on the world surface at a constant velocity. Simply put, the problem is the following: given measurements available to the observer, estimate the state of the target. Target state is defined to include the target's position and velocity; more precise definitions of these state qualities will be developed in Section 3.2.1. The wording of this problem naturally implies a state space mathematical formulation. Consequently, Kalman estimation techniques represent a very logical approach.

In formulating the state space equations, the choice of state variables and their associated coordinate systems is of prime importance. In this section, two sets of state variables will be introduced. The first set of state variables will include velocity components in Cartesian coordinates so that the nonlinearities inherent in the problem are obvious. There are many excellent classical primers on Kalman estimation theory, such as those by Gelb, Maybeck, and Zarchan and Musoff [87]–[89], but one thought from the fine textbook by Grewal and Andrews [90] succinctly states the necessary approach to forming these equations: "Think continuously. Act discretely."

3.2.1 State Equation

The state of the target will be defined first as the target's coordinates with respect to a global Cartesian coordinate system, and the components of the target's velocity with respect to each coordinate direction.

A fundamental assumption for this problem is that the target's motion is constrained to a two-dimensional surface in the world; therefore, two coordinate variables, x and y only, will be used most often to describe the target's position, with $\dot{x}$ and $\dot{y}$ denoting the first derivatives with respect to time of each coordinate variable. A further assumption is that

the target's acceleration values in the x and y directions are random values that fluctuate around zero. The continuous-time state equation for a target under these conditions is:

$$\underbrace{\begin{bmatrix} \dot{x} \\ \dot{y} \\ \ddot{x} \\ \ddot{y} \end{bmatrix}}_{\dot{\mathbf{x}}} = \underbrace{\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix}}_{\mathbf{F}} \underbrace{\begin{bmatrix} x \\ y \\ \dot{x} \\ \dot{y} \end{bmatrix}}_{\mathbf{x}} + \underbrace{\begin{bmatrix} 0 & 0 \\ 0 & 0 \\ 1 & 0 \\ 0 & 1 \end{bmatrix}}_{\mathbf{G}} \underbrace{\begin{bmatrix} w_{\ddot{x}} \\ w_{\ddot{y}} \end{bmatrix}}_{\mathbf{w}} \tag{3.18}$$

where $\ddot{x}$ and $\ddot{y}$ denote second derivatives with respect to time in the respective directions. In this model, the process noise is represented by a random perturbation to the target's acceleration, with components $w_{\ddot{x}}$ and $w_{\ddot{y}}$. The quantities $w_{\ddot{x}}(t)$ and $w_{\ddot{y}}(t)$ are therefore random processes, and can be described mathematically as *white noise* with zero mean; Brown and Hwang in their book [91, ch. 2] offer a very clear explanation of a random process defined in this manner. After condensing Equation (3.18) into vector-matrix form,

$$\dot{\mathbf{x}}(t) = \mathbf{F}\mathbf{x}(t) + \mathbf{G}\mathbf{w}(t) \tag{3.19}$$

let the vector $\mathbf{x}$ be the state vector and the vector $\mathbf{w}$ be the process noise vector.

So that it can be usable in a discrete-time estimation algorithm, the continuous-time state equation (3.18) must be transformed into a discrete-time difference equation:

$$\mathbf{x}[n+1] = \mathbf{\Phi}\mathbf{x}[n] + \mathbf{G}\mathbf{w}[n] \tag{3.20}$$

where matrix $\mathbf{G}$ is identical to that in Equation (3.18), and vectors $\mathbf{x}[n]$ and $\mathbf{w}[n]$ are sampled versions of their counterparts in Equation (3.18). The two quantities that must be derived from this new discrete-time form are the discrete-time state transition matrix $\mathbf{\Phi}[n]$, and the discrete-time process noise covariance matrix $\mathbf{Q}[n]$. The former is calculated in a

straightforward manner using the matrix exponential, yielding

$$\mathbf{\Phi} = e^{\mathbf{F}T_s} = \begin{bmatrix} 1 & 0 & T_s & 0 \\ 0 & 1 & 0 & T_s \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{3.21}$$

where T_s is the fundamental sampling interval of the discrete-time system.

Before calculating the process noise covariance, one must reconcile the fact that the theoretical continuous-time white-noise model has an infinite variance. The assumption that this theoretical continuous-time white noise is band-limited and also a stationary process results in the noise having a finite and constant variance, labeled σ_w^2. For this first four-state case where the state vector is constructed of positions and velocities of the target with respect to orthogonal coordinates in the world plane, a simplifying assumption was made that the random processes representing acceleration perturbations in the two directions were uncorrelated, and that the noise power for noise in each direction was equal, each with value σ_w^2. Note that models with a different set of state variables will be developed in Section 3.2.4 that are more realistically representative of a nonholonomic vehicle such as a ship moving on a surface.

Let the continuous-time process noise covariance matrix be denoted by $\mathbf{Q}_c$, and conform to the following definition:

$$\mathbf{Q}_c = E\left\{\mathbf{G}\mathbf{w}(t)\mathbf{w}^{\mathrm{T}}(t)\mathbf{G}^{\mathrm{T}}\right\} \tag{3.22}$$

which turns out to be a 4×4 diagonal matrix with σ_w^2 on the last two diagonal entries:

$$\mathbf{Q}_c = \begin{bmatrix} 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & \sigma_w^2 & 0 \\ 0 & 0 & 0 & \sigma_w^2 \end{bmatrix}, \tag{3.23}$$

where, again, σ_w^2 represents the noise power, also variance, of *band-limited* noise that is also a stationary process; therefore it has a finite and constant variance.

Assuming that $\mathbf{Q}_c$ is a constant matrix leads to an integral expression for computing discrete-time process noise covariance matrix $\mathbf{Q}[n]$. This matrix $\mathbf{Q}$ is computed as follows:

$$\mathbf{Q} = \int_0^{T_s} \mathbf{\Phi}(\tau)\mathbf{Q}_c\mathbf{\Phi}^{\mathrm{T}}(\tau)\, d\tau = \sigma_w^2 \begin{bmatrix} \frac{T_s^3}{3} & 0 & \frac{T_s^2}{2} & 0 \\ 0 & \frac{T_s^3}{3} & 0 & \frac{T_s^2}{2} \\ \frac{T_s^2}{2} & 0 & T_s & 0 \\ 0 & \frac{T_s^2}{2} & 0 & T_s \end{bmatrix}. \tag{3.24}$$

In practice, both matrices $\mathbf{\Phi}[n]$ and $\mathbf{Q}[n]$ can be calculated simultaneously using an efficient algorithm developed by Van Loan [92]. For the simulations described in Section 3.4, the Van Loan algorithm is implemented in the MATLAB language using intermediate matrices $\mathbf{M}$ and $\mathbf{N}$ to calculate square matrices $\mathbf{\Phi}$ and $\mathbf{Q}$. Each dimension of matrices $\mathbf{\Phi}$ and $\mathbf{Q}$ is of a size equal to the number of states of the system. Practitioners of estimation theory commonly use a lowercase letter n to represent this number of states; for example, see Gelb [87]. This usage is not to be confused with the use of lowercase n in this dissertation to represent the time index. The full code listing for this algorithm is shown in Appendix A.

3.2.2 Measurement Equation

The development of the measurement equation is somewhat less involved than the preceding derivation of the state equation because the measurement is inherently a discrete-time process. The actual quantities measured by the observer include its own position and orientation relative to a global coordinate system, as well as measurements of the image of the target in the image plane. For this first derivation, measurements of the image will consist only of the coordinates of the centroid of the image in the image plane. For subsequent derivations, measurements of image orientation and size will be added.

The measurement equation must relate the gathered measurements to the target state; however, for this problem, the relationship is nonlinear. The nonlinearity arises from the projection of a three-dimensional target on to a two-dimensional image plane. One method by which a linear relationship can be preserved between the measurements and the state vector is by using *pseudomeasurements*. The method of using pseudomeasurements and another method using a nonlinear transformation appear in the following example.

For this example, let there be a linear measurement equation involving a measurement vector $\mathbf{z}$, state vector $\mathbf{x}$, and measurement noise vector $\mathbf{v}$:

$$\mathbf{z} = \mathbf{H}\mathbf{x} + \mathbf{v} \tag{3.25}$$

where $\mathbf{H}$ is the measurement matrix. In this case, there is a simple linear relationship between the state vector and the measurement vector. If instead the relationship between state vector and measurement is *nonlinear*, then one of the two methods mentioned must be used.

In order to adapt Equation (3.25) to account for a nonlinear relationship between available measurements and target state, one approach is to develop some nonlinear transformation of the state vector, $\mathbf{h}(\mathbf{x})$, such that a nonlinear measurement equation can be formed thus:

$$\mathbf{z} = \mathbf{h}(\mathbf{x}) + \mathbf{v} \tag{3.26}$$

where $\mathbf{v}$ is a vector of additive measurement noise. In this case, a widely-used assumption is that the elements of $\mathbf{v}$ are normally distributed with zero mean.

In contrast to using a nonlinear transformation on the state vector, the term *pseudomeasurement* is used in the literature, for instance in the book by Zarchan and Musoff [89]. Another usage of this term is in the paper by Song [93], in which the term is used to indicate the result of an algebraic transformation on measurement vector $\mathbf{z}$. The transformation is performed to ensure that the right-hand side of Equation (3.26) is the sum of a linear function of $\mathbf{x}$ and a noise vector; for example:

$$\mathbf{g}(\mathbf{z}) = \mathbf{H}\mathbf{x} + \boldsymbol{\gamma}. \tag{3.27}$$

In this case, the vector $\mathbf{g}(\mathbf{z})$ is called the pseudomeasurement vector and $\boldsymbol{\gamma}$ is called the vector of pseudomeasurement noise. A different symbol has been chosen here to emphasize the difference between measurement noise $\mathbf{v}$ and pseudomeasurement noise $\boldsymbol{\gamma}$.

The advantage of Equation (3.27) over Equation (3.26) is that, by forming the pseudomeasurement in Equation (3.27), the nonlinearity is isolated on the left side of the equation. On the other hand, when the measurement equation of (3.27) is implemented, the probabil-

ity distribution function of the random variables in the pseudomeasurement noise vector $\boldsymbol{\gamma}$ most likely is no longer Gaussian as is the (assumed to be) zero-mean, normally-distributed perturbations of additive measurement noise vector $\mathbf{v}$. Having to use a measurement noise vector $\boldsymbol{\gamma}$ that is not Gaussian in the Kalman estimation equations is a disadvantage, since the Kalman estimation algorithm assumes that the distribution of measurement noise is Gaussian. One method to mitigate this disadvantage is explored in Section 3.2.7.

The Kalman estimation algorithm requires that the measurement noise covariance matrix $\mathbf{R} = E\left\{\mathbf{v}\mathbf{v}^{\mathrm{T}}\right\}$, or, in the case of pseudomeasurements, $\mathbf{R} = E\left\{\boldsymbol{\gamma}\boldsymbol{\gamma}^{\mathrm{T}}\right\}$ be calculated. In general, some characteristics of actual additive sensor noise $\mathbf{v}$ can be surmised from the estimated accuracy of on-board sensors, whereas pseudomeasurement noise $\boldsymbol{\gamma}$ cannot be deduced directly from on board sensor specifications. Therefore, measurement noise covariance matrix $\mathbf{R} = E\left\{\boldsymbol{\gamma}\boldsymbol{\gamma}^{\mathrm{T}}\right\}$ can be difficult to compute.

Amplifying this point, let $\mathbf{z}^{\star}$ represent the *perfect*, or uncorrupted measurement vector that corresponds to state $\mathbf{x}$ exactly, as

$$\mathbf{g}(\mathbf{z}^{\star}) = \mathbf{H}\mathbf{x} \tag{3.28}$$

related to actual measurement vector $\mathbf{z}$ by

$$\mathbf{z} = \mathbf{z}^{\star} + \mathbf{v}. \tag{3.29}$$

Subtracting Equation (3.28) from Equation (3.27) yields

$$\begin{aligned} \boldsymbol{\gamma}(\mathbf{z}^{\star}, \mathbf{v}) &= \mathbf{g}(\mathbf{z}) - \mathbf{g}(\mathbf{z}^{\star}) \\ &= \mathbf{g}(\mathbf{z}^{\star} + \mathbf{v}) - \mathbf{g}(\mathbf{z}^{\star}). \end{aligned} \tag{3.30}$$

The pseudomeasurement noise vector has now been written explicitly as a function of the uncorrupted measurement and the additive measurement noise.

Returning to the calculation of measurement noise covariance matrix $\mathbf{R}$, there are derivative-based and sampling-based methods to calculate this matrix. The derivative-based method will be discussed first, and the sampling-based method will be discussed in Section 3.2.7.

The first step in derivative-based approach to calculating the pseudomeasurement noise

covariance matrix (which will be given the symbol $\mathbf{C}_{\boldsymbol{\gamma}}$) is to view the additive measurement noise vector $\mathbf{v} \in \mathbb{R}^{\ell \times 1}$ as a small perturbation to the uncorrupted measurement vector $\mathbf{z}^\star \in \mathbb{R}^{\ell \times 1}$. Using the definition of the derivative for nonlinear transformation $\mathbf{g}(\mathbf{z}^\star) \in \mathbb{R}^{m \times 1}$ yields

$$\frac{\partial \mathbf{g}(\mathbf{z}^\star)}{\partial \mathbf{z}^\star}\mathbf{v} = \mathbf{g}(\mathbf{z}^\star + \mathbf{v}) - \mathbf{g}(\mathbf{z}^\star) \tag{3.31}$$

where the partial derivative has dimension $m \times \ell$.

Therefore, comparing the right-hand-side of Equation (3.30) with the right-hand-side of Equation (3.31), it is evident that the pseudomeasurement noise vector $\boldsymbol{\gamma}(\mathbf{z}^\star, \mathbf{v})$ is equal to the partial derivative multiplied by the additive noise vector, as:

$$\boldsymbol{\gamma}(\mathbf{z}^\star, \mathbf{v}) = \frac{\partial \mathbf{g}(\mathbf{z}^\star)}{\partial \mathbf{z}^\star}\mathbf{v}. \tag{3.32}$$

Because the uncorrupted measurement vector $\mathbf{z}^\star$ is unknown to the observer, the partial derivative can be approximated by evaluating it using the actual measurement vector $\mathbf{z}$ (assuming that the additive measurement noise vector $\mathbf{v}$ is small):

$$\boldsymbol{\gamma}(\mathbf{z}^\star, \mathbf{v}) \cong \left.\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right|_{\mathbf{z}} \mathbf{v}. \tag{3.33}$$

To compute the covariance matrix $\mathbf{C}_{\boldsymbol{\gamma}}$, one may start with the assumption that $\boldsymbol{\gamma}(\mathbf{z}^\star, \mathbf{v})$ is a random vector with zero mean. Indeed, the assumption has already been stated that the additive measurement noise vector $\mathbf{v}$ is presumed to have zero mean, and using Equation (3.30):

$$\boldsymbol{\gamma}(\mathbf{z}^\star, \mathbf{0}) = \mathbf{0}. \tag{3.34}$$

This last finding does not prove that pseudomeasurement noise vector $\boldsymbol{\gamma}(\mathbf{z}^\star, \mathbf{v})$ has zero mean, but does show that it has a value of zero whenever the value of the additive measurement noise vector $\mathbf{v}$ is zero. Using the definition for the covariance matrix, and for a given

measurement vector $\mathbf{z}$, the covariance matrix $\mathbf{C}_{\boldsymbol{\gamma}}$ can be computed as follows:

$$\begin{aligned}
\mathbf{C}_{\boldsymbol{\gamma}} &= E\left\{\left(\left.\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right|_{\mathbf{z}} \mathbf{v}\right)\left(\left.\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right|_{\mathbf{z}} \mathbf{v}\right)^{\mathrm{T}}\right\} \\
&= E\left\{\left(\left.\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right|_{\mathbf{z}}\right) \mathbf{v}\mathbf{v}^{\mathrm{T}}\left(\left.\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right|_{\mathbf{z}}\right)^{\mathrm{T}}\right\} \\
&= \left(\left.\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right|_{\mathbf{z}}\right) E\left\{\mathbf{v}\mathbf{v}\right\}^{\mathrm{T}}\left(\left.\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right|_{\mathbf{z}}\right)^{\mathrm{T}} \\
&= \underbrace{\left(\left.\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right|_{\mathbf{z}}\right)}_{m\times\ell} \underbrace{\mathbf{C}_{\mathbf{v}}}_{\ell\times\ell} \underbrace{\left(\left.\frac{\partial \mathbf{g}}{\partial \mathbf{z}}\right|_{\mathbf{z}}\right)^{\mathrm{T}}}_{\ell\times m}.
\end{aligned} \tag{3.35}$$

A suitable measurement error covariance matrix $\mathbf{C}_{\mathbf{v}}$ can be determined by analyzing the accuracy specification for the various on-board sensors. The matrix used in the Kalman estimation equations for the measurement noise covariance (traditionally labeled $\mathbf{R}$) is the pseudomeasurement noise covariance matrix just calculated, namely $\mathbf{C}_{\boldsymbol{\gamma}}$.

3.2.3 Sources of Uncertainty

Sections 3.2.1 and 3.2.2 have introduced the state and measurement equations, along with their associated noise covariance matrices $\mathbf{Q}$ and $\mathbf{R}$. This section will take a closer look at how the values for these two matrices are determined. Additionally, the significance of the state estimate error covariance matrix $\mathbf{P}$ will be examined.

Measurement uncertainty will be addressed first. Equation (3.17) contains the relationship that is fundamental in the examination of measurement uncertainty. The right-hand side actually contains all the measurements. Image plane measurements (u, v) are listed explicitly, and the other measurements of observer position and visual sensor orientation are contained in the matrix $\mathbf{M}^{-1}$. To address this last point, the definition of $\mathbf{M}$ is explicitly given from Equation (3.15):

$$\mathbf{M} = \begin{bmatrix} 0 & f & 0 & 0 \\ 0 & 0 & f & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \underbrace{\left[\begin{array}{cc:c} \multicolumn{2}{c:}{\multirow{2}{*}{$\mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda})$}} & \multirow{2}{*}{$-\mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda})^{\mathrm{g}}\mathbf{d}_o$} \\ & & \\ \hdashline 0 & 0 & 1 \end{array}\right]}_{3^{\mathrm{rd}}\text{ column removed}} \tag{3.36}$$

where ${}^{\mathrm{g}}d_o$ is the position of the observer in global coordinates, in other words the observer's GPS position, and $\mathbf{\Lambda}$ is the triplet of Euler angles of the observer's *visual sensor*. The relationship between the orientations of the visual sensor and observer's body is assumed to be fixed; for example, a camera at a fixed downward look angle from the observer's longitudinal axis.

The central problem, then, is to determine the effect of uncertainties in the variables on the right-hand side of Equation (3.17) on the resulting estimate of the target's position on the left-hand side of Equation (3.17). Section 3.2.2 described the derivative-based method for determining the effect of measurement uncertainties on the estimate of the target, and Section 3.2.7 will present a sampling-based method to do the same. In either case, some basic assumptions need to be made about the uncertainty in each individual measured quantity. Assumed values for one standard deviation of uncertainty for various measured quantities are listed in Table 3.1. The values listed in Table 3.1 represent an attempt to model complex sensor error dynamics of GPS and of micro-electro-mechanical systems (MEMS) accelerometers using simple Gaussian random variables. The following paragraph will explore this idea in more detail.

In particular, much research has been done on error models for GPS receivers. The authoritative volume from Parkinson and Spilker [94] contains extensive information on various error models. Overall, the horizontal error from GPS is usually represented by a non-stationary stochastic process. To keep the GPS error model fairly simple for the simulation in this chapter, values for GPS uncertainty using a stationary Gaussian random process were chosen after reviewing nominal values from Biezad [95, Tab. 6.1, p. 112]. Biezad listed one standard deviation of horizontal error to be 40 m, and one standard deviation of vertical error to be 50 m for the standard positioning service (SPS), assuming that the SPS was activated and degrading the navigation calculations on the user equipment. Because the Selective Availability function of GPS was disabled by U.S. President Bill Clinton's executive order in 2000, and has not since been reactivated, the values for one standard deviation of horizontal and vertical error were reduced to 10 m horizontal, and 20 m vertical, respectively. These modified values are the ones listed in Table 3.1.

Modern MEMS sensors that are found in devices ranging from automobiles to cellular telephones also have sensor noise characteristics that have been widely studied. One recent

survey paper consulted for this work was Mohd-Yasin [96], which offered an overview of electronic, mechanical, and thermal noise models for MEMS sensors. Instead of using a complex noise model, simple values to use for Euler angle uncertainty for the simulation were chosen after consulting datasheets of various MEMS-based angular rate sensors.

The last sensor to be considered was the visual sensor itself, and the values for uncertainty in the image position coordinates was based on the size of one pixel on the image plane of a common miniature video recording system, the GoPro camera. For all sensors being considered, the uncertainty values thus chosen were assigned as the value of one standard deviation (1σ) of the random variables that would be used to represent the uncertainty in the vector of measurement noise $\mathbf{v}$ as discussed in Section 3.2.2.

Table 3.1: Presumed values of sensor estimation errors. In this case, errors are assumed to be normally-distributed random variables having standard deviations equal to the values shown.

Parameter	Value of 1σ
GPS error North/East	10 m
GPS error Down	20 m
Euler angle error	2°
image position ${}^{\mathrm{i}}x$	9 µm
image position ${}^{\mathrm{i}}y$	8.94 µm

In particular, for the derivative-based method, the standard deviations so chosen could then be used to construct the measurement error covariance matrix, $\mathbf{C}_{\mathbf{v}}$, in Equation (3.35). More difficult to calculate are the vector gradients of Equation (3.35), $\frac{\partial \mathbf{g}}{\partial \mathbf{z}}$, where the vector function $\mathbf{g}$ represents the nonlinear transformation embodied in multiplication by matrix $\mathbf{M}^{-1}$ in Equation (3.17). In practice, these derivatives were calculated using a symbolic differentiation algorithm, for example the symbolic mathematics package in MATLAB.

One last thought on measurement uncertainty: some combinations of position and orientation of the observer relative to the target may cause the transformation represented by (3.17) to become ill-conditioned; in other words, small errors in the measurements may cause large errors in the estimates.

Process uncertainty addresses errors in the state-space model incorporated in the Kalman estimator as described in Section 3.2.1. The fundamental assumptions made when for-

mulating the state-space model were that the target's position was constrained to a 2D plane in the global coordinate system (${}^{g}z_T = 0$), and that the target's motion was along a straight-line course at constant speed. The first assumption would be violated slightly if the wave-motion effects on a vessel on the sea surface were taken into account. The second assumption is more likely to be violated in a live flight test scenario simply due to the difficulty in maintaining precisely constant course and speed for a vessel underway.

For this reason, some *a priori* estimate of the process uncertainty must be provided to the estimator. Table 3.2 contains the values that were used for the simulations described in Section 3.4. The values chosen for the process uncertainty were selected to be representative of a vessel attempting to maintain constant course and speed.

Table 3.2: Presumed values of process model errors. Errors are assumed to be normally-distributed random variables having standard deviations equal to the values shown.

Parameter	Value of 1σ
Target turn rate uncertainty	0.4 °/s
Target acceleration uncertainty	0.05 m/s^2

State estimate uncertainty, which is the final category of uncertainty, is primarily a result of the estimation process, although it does depend on an *a priori* initial value. In the case of an airborne ADS seeking a moving surface target, the initial uncertainty in the target's position may be rather high—on the order of kilometers—if the target has not been sighted immediately prior to ADS deployment. Initial uncertainty in target speed is naturally limited by the range of realistic ship speeds. Initial uncertainty in course is bounded by the very nature of the angular measure used. The amounts of presumed error in the initial estimates of the target are shown in Table 3.3.

Table 3.3: Errors in the initial state estimates. These are assumed to be normally-distributed random variables having standard deviations equal to the values shown.

Parameter	Value of 1σ
North/East position error	500 m
Speed error	2.57 m/s
Course error	20°

Once the estimator is operating, a state estimation error covariance matrix $\mathbf{P}$ is computed at every time step. Because the first two elements of the state vector are the estimated North and East coordinates of the target, one very useful aspect of this matrix $\mathbf{P}$ is that the upper-left 2×2 submatrix can be used to generate an elliptical two-dimensional contour in global coordinates representing uncertainty in the target position estimate. The eigenvalues of this 2×2 submatrix represent the directions of the principal axes of the ellipse, and the square roots of the eigenvalues can be scaled to represent the length of one standard deviation along these principal axes.

3.2.4 Alternate State Formulation and Nonlinear State Update

The original formulation of the state vector described in Section 3.2.1 consisting of two coordinates each of position and velocity was chosen simply for convenience and ease of rapid prototyping with some initial algorithms. The state-space model of the surface target is more naturally described in terms of its 2D position and its speed and course. As will be shown, this new state vector formulation in terms of (x, y, V, ψ_T) allows an additional measurement from the image plane to be incorporated.

To adapt the state equation for this new state vector, the vector-matrix expression must be changed from Equation (3.19) to one in which the state derivatives are calculated using a nonlinear function:

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t)) + \mathbf{G}\mathbf{w}(t). \tag{3.37}$$

The nonlinear vector function $\mathbf{f}(\mathbf{x})$ used to compute the state derivatives is fairly straightforward and is shown here in terms of the derivatives of the individual states:

$$\dot{x} = V \cos \psi_T \tag{3.38}$$

$$\dot{y} = V \sin \psi_T \tag{3.39}$$

$$\dot{V} = 0 \tag{3.40}$$

$$\dot{\psi}_T = 0. \tag{3.41}$$

For what is commonly known as the EKF (herein called the extended Kalman estimator), there is still a need to use a discrete-time transition matrix $\mathbf{\Phi}$, which is formed from a

continuous-time state transition matrix $\mathbf{F}$. In this case of the extended Kalman estimator, the Jacobian of nonlinear vector function $\mathbf{f}(\mathbf{x})$ must be used, as:

$$\mathbf{F} = \left.\frac{\partial \mathbf{f}(\mathbf{x})}{\partial \mathbf{x}}\right|_{\mathbf{x}=\hat{\mathbf{x}}} = \begin{bmatrix} \frac{\partial \dot{x}}{\partial x} & \frac{\partial \dot{x}}{\partial y} & \frac{\partial \dot{x}}{\partial V} & \frac{\partial \dot{x}}{\partial \psi_{\mathbf{T}}} \\ \frac{\partial \dot{y}}{\partial x} & \frac{\partial \dot{y}}{\partial y} & \frac{\partial \dot{y}}{\partial V} & \frac{\partial \dot{y}}{\partial \psi_{\mathbf{T}}} \\ \frac{\partial \dot{V}}{\partial x} & \frac{\partial \dot{V}}{\partial y} & \frac{\partial \dot{V}}{\partial V} & \frac{\partial \dot{V}}{\partial \psi_{\mathbf{T}}} \\ \frac{\partial \dot{\psi}_{\mathbf{T}}}{\partial x} & \frac{\partial \dot{\psi}_{\mathbf{T}}}{\partial y} & \frac{\partial \dot{\psi}_{\mathbf{T}}}{\partial V} & \frac{\partial \dot{\psi}_{\mathbf{T}}}{\partial \psi_{\mathbf{T}}} \end{bmatrix}_{\mathbf{x}=\hat{\mathbf{x}}} \tag{3.42}$$

where $\hat{\mathbf{x}}$ is the current estimate of the state vector, and the partial derivatives in the Jacobian matrix are calculated from the state derivatives of Equations (3.38) to (3.41):

$$\frac{\partial \dot{x}}{\partial V} = \sin \psi_T \qquad \frac{\partial \dot{x}}{\partial \psi_{\mathbf{T}}} = V \cos \psi_T \tag{3.43}$$

$$\frac{\partial \dot{y}}{\partial V} = \cos \psi_T \qquad \frac{\partial \dot{y}}{\partial \psi_{\mathbf{T}}} = -V \sin \psi_T . \tag{3.44}$$

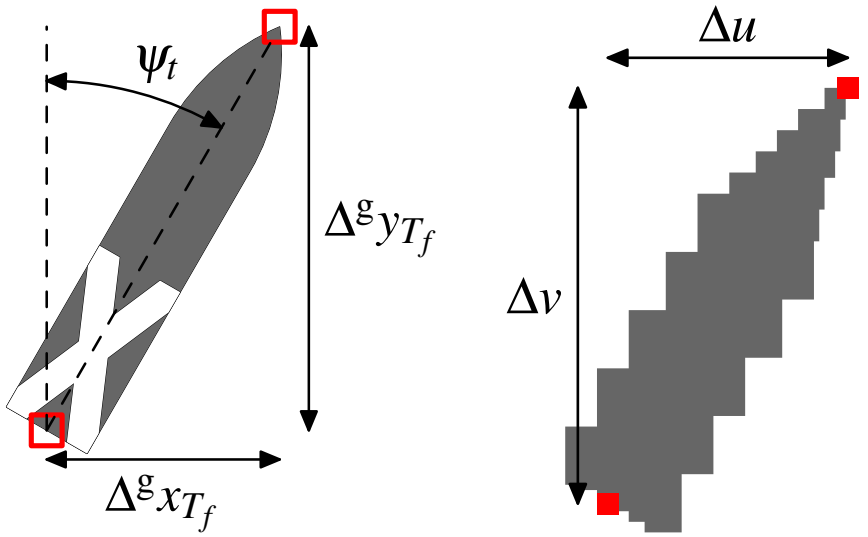
The remaining partial derivatives in the Jacobian matrix (3.42) not specified in Equations (3.43) and (3.44) are zero.

With this change, process noise is now assumed to enter to the model on the derivative of target speed V, which is target acceleration $\dot{V}$, and on the derivative of target course ψ_T, which is target turn rate $\dot{\psi}_T$. The continuous-time process noise covariance matrix $\mathbf{Q}_c$ is a 4×4 diagonal matrix with acceleration noise and target turn-rate noise as the last two entries:

$$\mathbf{Q}_c = \begin{bmatrix} 0 & & & \\ & 0 & & \\ & & \sigma^2_{\dot{V}} & \\ & & & \sigma^2_{\dot{\psi}_T} \end{bmatrix} . \tag{3.45}$$

The Van Loan algorithm first mentioned in Section 3.2.1 is then employed to calculate the discrete-time state transition and process noise covariance matrices $\mathbf{\Phi}$ and $\mathbf{Q}$ that are in turn used to calculate state estimation error covariance matrix $\mathbf{P}$.

The change to the measurement equation that is caused by using this alternate formulation of the vector is the introduction of an additional possible measurement from the image plane information. The assumption that the target's *motion* is constrained to a 2D surface



(a) Actual target in global coordinate frame.

(b) Image of target in image plane.

Figure 3.3: Target features in global frame (a) and image features on image plane (b). Bow and stern features on the target in the global coordinate frame are related to these same features on the image of the target in the image plane.

in the world must be again refined to become an assumption that the target *itself* is an elongated object constrained to lie horizontally on the world's surface and that the target's motion is in the direction of its own longitudinal axis. The target's longitudinal axis would, of course, be constrained to the 2D surface within the global coordinate system. These assumptions are followed to a first order of approximation by a vessel underway on calm seas.

Under these conditions, the orientation of the target on a 2D surface in the world (hence its course, ψ_T), may be inferred by the orientation ψ_i of the elongated image on the image plane. In the following, it will be assumed that the ends of the target's image, the bow and stern features, are able to be extracted flawlessly from the target's image. The additional measurements from the image will then be the horizontal and vertical differences on the image plane between the bow and stern features.

Consider the depiction in Figure 3.3 of the target with bow and stern points identified in the global frame (Figure 3.3a). Let $\Delta^{\mathrm{g}} x_{T_f}$ and $\Delta^{\mathrm{g}} y_{T_f}$ be the difference in global coordinates

between the bow and stern features of the target. Therefore, there are two positions of interest for the target that are contained in the following vectors:

$$ {}^{\mathrm{g}}\mathbf{p}_{T_1} = \begin{bmatrix} {}^{\mathrm{g}}x_{T_f} \\ {}^{\mathrm{g}}y_{T_f} \\ 0 \\ 1 \end{bmatrix} \qquad \text{and} \qquad {}^{\mathrm{g}}\mathbf{p}_{T_2} = \begin{bmatrix} {}^{\mathrm{g}}x_{T_f} + \Delta^{\mathrm{g}}x_{T_f} \\ {}^{\mathrm{g}}y_{T_f} + \Delta^{\mathrm{g}}y_{T_f} \\ 0 \\ 1 \end{bmatrix} \tag{3.46} $$

where $({}^{\mathrm{g}}x_{T_f}, {}^{\mathrm{g}}y_{T_f})$ are the coordinates of one feature, such as the bow of the target ship. The third coordinate of each vector is zero because the entire target ship, including bow and stern, must line on the world surface plane. Each of the vectors in Equation (3.46) thus loses its third element, and then each serves as input to Equation (3.16) to compute image plane coordinates (u_1, v_1) and (u_2, v_2) corresponding to the bow and stern locations of the image.

$$ \begin{bmatrix} w_1 u_1 \\ w_1 v_1 \\ w_1 \end{bmatrix} = \mathbf{M}(\mathbf{\Lambda}, {}^{\mathrm{g}}\mathbf{d}) \begin{bmatrix} {}^{\mathrm{g}}x_{T_f} \\ {}^{\mathrm{g}}y_{T_f} \\ 1 \end{bmatrix} \tag{3.47} $$

$$ \begin{bmatrix} w_2 u_2 \\ w_2 v_2 \\ w_2 \end{bmatrix} = \mathbf{M}(\mathbf{\Lambda}, {}^{\mathrm{g}}\mathbf{d}) \begin{bmatrix} {}^{\mathrm{g}}x_{T_f} + \Delta^{\mathrm{g}}x_{T_f} \\ {}^{\mathrm{g}}y_{T_f} + \Delta^{\mathrm{g}}y_{T_f} \\ 1 \end{bmatrix}. \tag{3.48} $$

In Equations (3.47) and (3.48), the notation $\mathbf{M}(\mathbf{\Lambda}, {}^{\mathrm{g}}\mathbf{d})$ emphasizes that $\mathbf{M}$ depends only on the location and orientation of the observer, not the target. Also, notice that the scale factors w_1 and w_2 have been applied to the image plane coordinates instead of the global coordinates. These scale factors depend on the distance between the target and the focal point. Consequently, w_1 and w_2 have different values, because, in the majority of cases, the target's bow and stern will be at different distances from the observer.

The coordinate vector for the stern feature consists of two components, which are the bow

feature position, and the differences between bow and stern:

$$\begin{bmatrix} w_2 u_2 \\ w_2 v_2 \\ w_2 \end{bmatrix} = \mathbf{M}(\mathbf{\Lambda}, {}^{\mathrm{g}}\mathbf{d}) \left(\begin{bmatrix} {}^{\mathrm{g}}x_{T_f} \\ {}^{\mathrm{g}}y_{T_f} \\ 1 \end{bmatrix} + \begin{bmatrix} \Delta^{\mathrm{g}}x_{T_f} \\ \Delta^{\mathrm{g}}y_{T_f} \\ 0 \end{bmatrix} \right) \tag{3.49}$$

where the first term on the right-hand side is simply the set of coordinates for the bow feature:

$$\begin{bmatrix} w_2 u_2 \\ w_2 v_2 \\ w_2 \end{bmatrix} = \begin{bmatrix} w_1 u_1 \\ w_1 v_1 \\ w_1 \end{bmatrix} + \mathbf{M}(\mathbf{\Lambda}, {}^{\mathrm{g}}\mathbf{d}) \begin{bmatrix} \Delta^{\mathrm{g}}x_{T_f} \\ \Delta^{\mathrm{g}}y_{T_f} \\ 0 \end{bmatrix}. \tag{3.50}$$

The third row of vector-matrix Equation (3.50) indicates that scale factor w_2 is equal to w_1 plus a linear combination of coordinate difference values $\Delta^{\mathrm{g}}x_{T_f}$ and $\Delta^{\mathrm{g}}y_{T_f}$. The scale factor and also the image plane coordinates can then be handled in the same manner as the global coordinates of target features: by using an anchor value, which is the bow feature image, and then computing the differences in coordinates as follows:

$$w_2 = w_1 + \Delta w \tag{3.51}$$

$$u_2 = u_1 + \Delta u \tag{3.52}$$

$$v_2 = v_1 + \Delta v. \tag{3.53}$$

The differences in image plane coordinates between the bow and stern features will be labeled Δu and Δv. Note that differences Δu and Δv can be expressed in units of pixels or units of length; however, in the following expressions, Δu and Δv will be assumed to be in units of length.

The substitution of Equations (3.51), (3.52), and (3.53) into Equation (3.48) leads to a single expression in terms of the anchor coordinates and differences:

$$\begin{bmatrix} (w_1 + \Delta w)(u_1 + \Delta u) \\ (w_1 + \Delta w)(v_1 + \Delta v) \\ w_1 + \Delta w \end{bmatrix} = \mathbf{M}(\mathbf{\Lambda}, {}^{\mathrm{g}}\mathbf{d}) \begin{bmatrix} {}^{\mathrm{g}}x_{T_f} + \Delta^{\mathrm{g}}x_{T_f} \\ {}^{\mathrm{g}}y_{T_f} + \Delta^{\mathrm{g}}y_{T_f} \\ 1 \end{bmatrix} \tag{3.54}$$

The left-hand side of Equation (3.54) expands, and the terms containing only incremental

difference values such as $\Delta w \Delta u$ subsequently vanish:

$$\begin{bmatrix} w_1 u_1 \\ w_1 v_1 \\ w_1 \end{bmatrix} + \begin{bmatrix} w_1 \Delta u + \Delta w u_1 + \Delta w \Delta u \\ w_1 \Delta v + \Delta w v_1 + \Delta w \Delta v \\ \Delta w \end{bmatrix} = \mathbf{M}(\mathbf{\Lambda}, {}^{\mathrm{g}}\mathbf{d}) \begin{bmatrix} {}^{\mathrm{g}}x_{T_f} \\ {}^{\mathrm{g}}y_{T_f} \\ 1 \end{bmatrix} + \mathbf{M}(\mathbf{\Lambda}, {}^{\mathrm{g}}\mathbf{d}) \begin{bmatrix} \Delta^{\mathrm{g}}x_{T_f} \\ \Delta^{\mathrm{g}}y_{T_f} \\ 0 \end{bmatrix} \tag{3.55}$$

The first term on both the left and right sides are equal to each other according to Equation (3.47), and thus also vanish.

The third row of Equation (3.55) yields a solution for the change in scale factor:

$$\Delta w = M(3,1)\Delta^{\mathrm{g}}x_{T_f} + M(3,2)\Delta^{\mathrm{g}}y_{T_f} \tag{3.56}$$

where a MATLAB-like notation indicates that the first term on the right-hand side is the third row, first column scalar value in the matrix $\mathbf{M}$ multiplied by the global coordinate difference in the x direction. Equation (3.56) supplies the definition for Δw for the first two rows of Equation (3.55), leading to an expression relating the horizontal and vertical differences on the image plane between bow and stern features, and the corresponding differences on a plane in global coordinates for the target's actual bow and stern. The resulting expression is:

$$\begin{bmatrix} w\Delta u \\ w\Delta v \end{bmatrix} = \underbrace{\left\{ \mathbf{M}(1:2,1:2) - \begin{bmatrix} u \\ v \end{bmatrix} \begin{bmatrix} M(3,1)M(3,2) \end{bmatrix} \right\}}_{\mathbf{S}} \begin{bmatrix} \Delta^{\mathrm{g}}x_{T_f} \\ \Delta^{\mathrm{g}}y_{T_f} \end{bmatrix}. \tag{3.57}$$

In Equation (3.57), again, a MATLAB-like notation is used to indicate a submatrix of $\mathbf{M}$ consisting of the first two rows and first two columns ($\mathbf{M}(1:2,1:2)$), and also the individual scalar elements of the matrix $\mathbf{M}$, such as the third row, first column ($M(3,1)$). Also in Equation (3.57), w is a scale factor for homogeneous coordinates, and $\Delta^{\mathrm{g}}x_{T_f}$ and $\Delta^{\mathrm{g}}y_{T_f}$ are meant to convey changes in global coordinates of *target features*, namely the difference between bow and stern positions.

As was the case for Equation (3.15), this expression is again almost, but not quite, the one that is needed. The measurements available to the estimator are the *image* plane differences,

$(\Delta u, \Delta v)$, so what is needed are the global coordinate differences in terms of the image plane differences. The 2×2 matrix indicated as $\mathbf{S}$ in Equation (3.57) must be inverted to obtain:

$$\begin{bmatrix} \Delta^{\mathrm{g}} x_{T_f} \\ \Delta^{\mathrm{g}} y_{T_f} \end{bmatrix} = \mathbf{S}^{-1} \begin{bmatrix} w\Delta u \\ w\Delta v \end{bmatrix}. \tag{3.58}$$

The target's computed course in the global coordinate system then is simply

$$\psi_T = \arctan\left(\frac{\Delta^{\mathrm{g}} x_{T_f}}{\Delta^{\mathrm{g}} y_{T_f}}\right). \tag{3.59}$$

Putting it all together under this alternate state vector formulation, the measurement equation becomes

$$\mathbf{g}(\mathbf{z}) = \mathbf{H}\mathbf{x} + \boldsymbol{\gamma} \tag{3.60}$$

where

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad \mathbf{x} = \begin{bmatrix} x \\ y \\ V \\ \psi_T \end{bmatrix}. \tag{3.61}$$

Under this new formulation, the full 10×1 measurement vector $\mathbf{z}$ now consists of three coordinates of observer position, three Euler angles of the observer's visual sensor, coordinates (u, v) of the target centroid on the image plane, and $(\Delta u, \Delta v)$ measurements of the slope of the elongated target image in the image plane. The nonlinear transformation $\mathbf{g}(\mathbf{z})$ transforms the 10×1 measurement vector $\mathbf{z}$ into a 3×1 vector of estimates of the target's two position coordinates and course. This nonlinear transformation is comprised of Equation (3.17) for the first two (position) elements of the 3×1 vector, and Equations (3.58) and (3.59) for the third element of the 3×1 vector, target course.

This additional measurement necessitates an additional step in the calculation of measure-

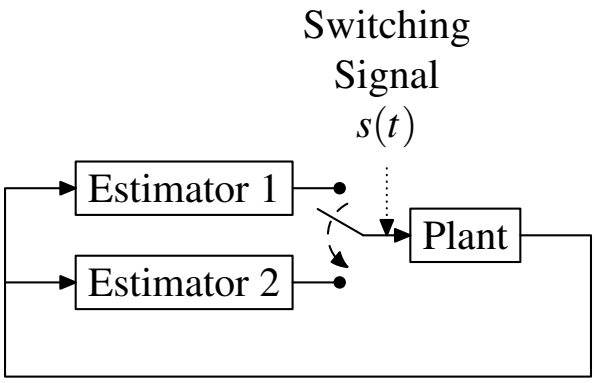


Figure 3.4: Motion estimation subsystem switching diagram. This block diagram of a switching scheme between two target motion estimators is reproduced from the work of L. Ma [97].

ment error. For the derivative-based method described in Section 3.2.2, the vector gradient of Equation (3.35), $\frac{\partial \mathbf{g}}{\partial \mathbf{z}}$, still needs to be calculated; however, in this case the first two rows are taken from the computation of the gradient of matrix **M**, and the last row is taken from the computation of the gradient of matrix **S**.

3.2.5 Epipolar Geometry and the Out-of-Frame Condition

Several research teams conducting target tracking experiments with UASs, for example, those at BYU and at NPS, have addressed the problem of an airborne observer tracking a surface moving target for the case that the target is not always in the field of view of the observer. Along these lines, the works of Hespanha [81] and of Dobrokhodov [82] were mentioned in Section 2.6.3. This work was continued by L. Ma [97] in the context of an airborne observer tracking a target on the surface moving along a road. While the application of Ma's work is different than the problem at hand, which is an airborne ADS tracking a target on the surface as a landing platform, Ma's treatment of the out-of-frame condition deserves special attention. In that work, the Boolean switching signal $s(t)$ is true if the target is in frame, false if the target is out of frame. This switching signal is used to choose between two target motion estimators. A conceptual block diagram of this switching scheme is shown in Figure 3.4.

One reason that a different approach was taken for the problem at hand was that Ma's algorithm, designed for a fixed-wing aircraft as the observer, depended on an upper bound of the duration of the brief instability, which is the duration that the target is out of frame. Un-

fortunately, the term *brief instability* does not quite describe the situation when the observer is mounted on the payload of a swinging parafoil in flight, and the question is whether a moving surface target remains in its field of view. Data and video from the flight tests described in Appendix B suggest that, in the case of an airborne, parafoil-mounted observer tracking a moving target on the Earth's surface, that the target would exit and re-enter the field of view many times and remain out-of-frame for significant durations of time.

Interestingly, an advantage is gained from casting this situation in terms of multiple-view geometry. Under this formulation, the instants when the target exits the field of view, and when it re-enters the field of view can be characterized as two views of the same scene. Of course, the target has moved between these two instants of time, but with the help of the assumption that the target's motion has constant course and speed (along with one other assumption), the second view of the target (entering frame) can be properly shifted in time so that it coincides with the first view. This method is similar to what a ship's navigator would call a *running fix*: sighting two lines of position (LOPs) at different times, then advancing or retarding in time one LOP so that it coincides with the other.

The second assumption mentioned above that is needed is the assumption that the target image can be condensed to a single point, for example the centroid of the target, so that the two views are of the very same point. Consider two views of the same point target depicted in Figures 3.5a and 3.5b. This depiction is of two separate time instants at which the observer positions and orientations with respect to global reference frame $\{g\}$ are $(\mathbf{d}_1, \Lambda_1)$ and $(\mathbf{d}_2, \Lambda_2)$. The target is out of the field of view of the observer between these two time instants, and the distance that the target has traveled over this duration is $\Delta^{\mathrm{g}}\mathbf{p}_T$. In Figure 3.5b, the second observation has been shifted by the movement $\Delta^{\mathrm{g}}\mathbf{p}_T$ of the target during the time that the target was out of view. This calculation is equivalent to retarding the second observation in time because the velocity of the target is assumed to be constant. Thus, Figure 3.5b depicts a situation where there are effectively two views of the *same* target.

Once the two-view geometry is characterized as in Figure 3.5, the so-called *epipolar constraint* can be invoked; a thorough and rigorous treatment of which can be found in Y. Ma [98]. The epipolar constraint stipulates that the lines of sight $\mathbf{v}_1$ and $\mathbf{v}_2$ must be coplanar with a line joining the two viewpoints. In Figure 3.5b, vector $\mathbf{c}$ lies along the line

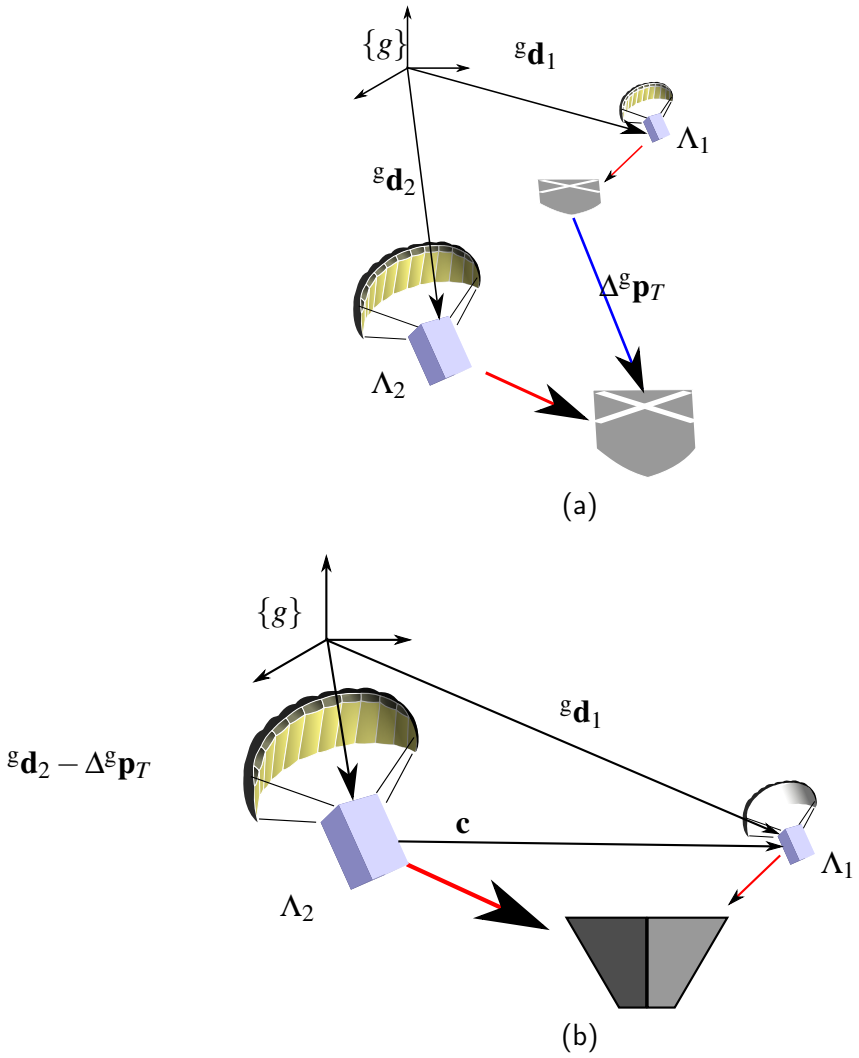


Figure 3.5: Two views of the target. Actual observations of the target are separated in time and space (a). Assuming constant target motion, the two views can be brought together (b).

joining the two viewpoints (focal points); therefore, the constraint is

$$\mathbf{v}_1 \cdot (\mathbf{c} \times \mathbf{v}_2) = 0 \tag{3.62}$$

where $\mathbf{v}_1$ and $\mathbf{v}_2$ are vectors representing lines of sight from each of the two focal points and pointing at the target. A mathematical statement of the condition that all three vectors in Equation (3.62) are co-planar is that their scalar triple product is zero.

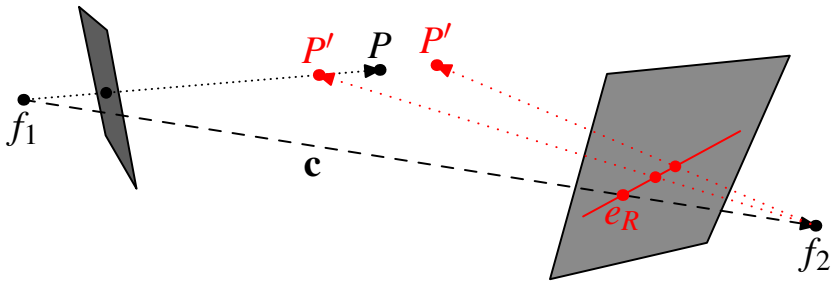


Figure 3.6: Epipolar geometry. Two views of the object at point P are made from focal points f_1 and f_2. Given the known line of sight from f_1 to P, other possible locations P' of the object are shown in red with associated image locations on the right-hand image plane.

Figure 3.6 contains an illustration of the geometry of the epipolar constraint. Two observers at focal points f_1 and f_2 observe an object located at point P. Baseline vector $\mathbf{c}$ connects the focal points. Given that the image location shown on the left-hand image plane is correct, then, from the view point of the observer on the left, the object must lie along the line of sight from f_1 through the image. Alternative object locations P' are shown. From the view point of the observer on the right, given the known line of sight from the left-hand observer, then all possible object locations are constrained to lie along a line through point e_R, which is called an *epipole*, and which is depicted in the right-hand image plane.

Returning to the two-view problem at hand, the epipolar constraint provides a novel method to measure the image on the image frame at the instant that the target returns to view. In Figure 3.7, the image plane is shown at the instant that the target returns to view, with the ship symbol representing the centroid of the target. An epipolar line in this image plane has been established by the last in-view image on a previous image plane, just prior to the instant when the target exited view. A scalar measurement can be made whose value is the distance between the target centroid, at the instant that the target returns to view, and the epipolar line. Let this measurement be called s'.

In Figure 3.8, both image planes are shown together, with the left-hand image plane representing last-in-view (target exiting frame), and the right-hand image plane representing first-in-view, or target returning to the frame. The geometry has been contrived to be simple in Figure 3.8, where f_1 and f_2 are the focal points of the two image planes, and $\mathbf{c}$ is the vector along the line joining the two focal points. Point P is the target's centroid position at the instant of leaving the frame, and it is co-planar with line $\mathbf{c}$ in a perfectly horizontal

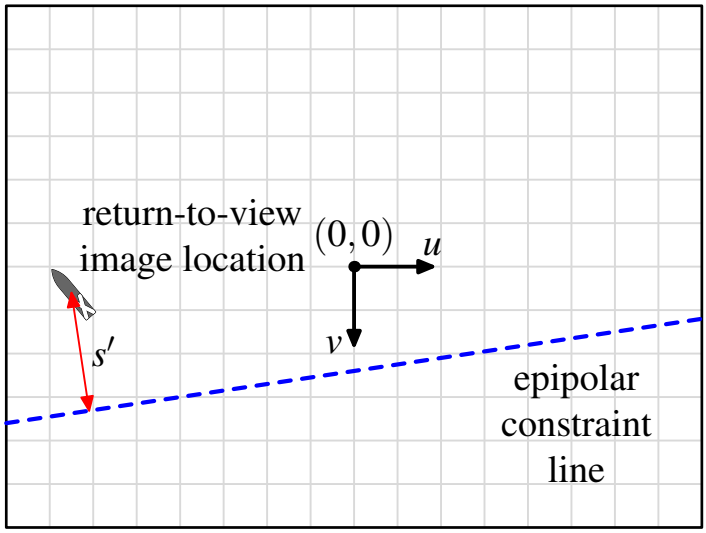


Figure 3.7: Epipolar constraint scalar distance. The distance between the return-to-view image and the epipolar constraint line of the two views is used to formulate additional measurement.

plane in this depiction. Point P' represents the target's centroid position at the instant of re-entering the frame. Ideally, because the returning-to-frame view has been shifted in time to coincide with the leaving frame view, points P and P' should also be coincident. Figure 3.8 depicts the case where P and P' are not coincident, and scalar distance s' is shown as the distance in the image plane between the aforementioned epipolar constraint line, and the intersection point of the image plane itself and the line of sight between f_2 to P'. Note that $\mathbf{c} \times \mathbf{v}_2$ is normal to the horizontal plane mentioned above; therefore, $\mathbf{v}_1 \cdot (\mathbf{c} \times \mathbf{v}_2)$ is a projection on this normal.

In practice, vector $\mathbf{c}$ that connects f_1 and f_2, the focal points of the two image planes, contains components due to both the observer's motion and also due to the target's motion; specifically:

$$^{\mathrm{g}}\mathbf{c} = {}^{\mathrm{g}}\mathbf{d}_1 - ({}^{\mathrm{g}}\mathbf{d}_2 - \Delta^{\mathrm{g}}\mathbf{p}_T) \tag{3.63}$$

where $\Delta^{\mathrm{g}}\mathbf{p}_T$ is a vector representing the change in position of the target during the time interval between the first image and the second image, and $^{\mathrm{g}}\mathbf{c}$ denotes explicitly that the baseline vector is expressed in global coordinates. The vector $\Delta^{\mathrm{g}}\mathbf{p}_T$ is *subtracted* from the position vector $^{\mathrm{g}}\mathbf{d}_2$ of the observer at the time at which the target returns to view because this second observation is being retarded to coincide with the first observation. For imple-

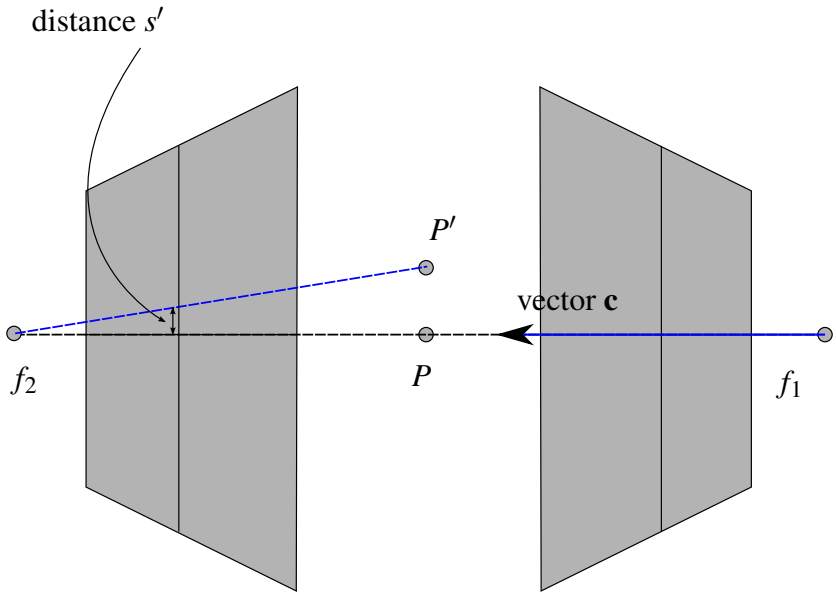


Figure 3.8: Epipolar constraint with both image planes. When the target reappears at location P', distance s' is the distance between the location of the image of P' and the epipolar constraint line.

mentation, the scalar distance s' is split into two components by splitting the baseline vector $\mathbf{c}$ into two components: one depending only on observer motion, and the other depending only on target motion, as in the following expression:

$$^{g}\mathbf{c} = \underbrace{^{g}\mathbf{d}_1 - {}^{g}\mathbf{d}_2}_{^{g}\mathbf{c}_1} + \underbrace{\Delta^{g}\mathbf{p}_T}_{^{g}\mathbf{c}_2}, \tag{3.64}$$

which, following from the definition of the epipolar constraint, Equation (3.62), allows:

$$\mathbf{v}_1 \cdot (\mathbf{c} \times \mathbf{v}_2) = 0 = s' \tag{3.65}$$

$$\mathbf{v}_1 \cdot [(\mathbf{c}_1 + \mathbf{c}_2) \times \mathbf{v}_2] = 0 \tag{3.66}$$

$$\underbrace{\mathbf{v}_1 \cdot (\mathbf{c}_1 \times \mathbf{v}_2)}_{s} = \underbrace{\mathbf{v}_1 \cdot (\mathbf{v}_2 \times \mathbf{c}_2)}_{\hat{s}} \tag{3.67}$$

Thus, the predicted scalar measurement $\hat{s}$ is calculated based on the component containing the target's motion, and the scalar measurement s itself is calculated from component $\mathbf{c}_1$

that depends on the observer's motion. The vector equation in Equation (3.67) requires that all vectors be expressed in the same coordinate system; however, the baseline vector $\mathbf{c}$ and the lines of sight, vectors $\mathbf{v}_1$ and $\mathbf{v}_2$, are defined with respect to three different coordinate frames. Both components of baseline vector $\mathbf{c}$ come most naturally in the global coordinate frame as the differences in the two observer positions, $\mathbf{c}_1$, and the assumed motion of the target, which is $\mathbf{c}_2$. Line of sight vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ are actually defined in reference frames slightly different from the image plane reference frame defined in Section 3.1.3.

3.2.6 Derivation of the Epipolar Constraint Measurement

Consider the depiction in Figure 3.9 of two observers viewing the same target; however, in this case, the two image coordinate frames are centered at their respective focal points. The origins of image plane reference frames $\{i_1\}$ and $\{i_2\}$ are now coincident with the respective focal points f_1, and f_2, but have the same orientation with each x axis aligned with the optical axis and passing through the center of the image plane. Each line of sight is a ray originating at the focal point, and passing through both the image and the target. The coordinates of the target are not known with certainty in this frame, but the coordinates of the image are known, because the image is located exactly one focal length f from the new origin of this frame. Thus, components $[fuv]^{\mathrm{T}}$ constitute the line of sight vector in each image plane reference frame.

The vector cross-products in Equation (3.67) must have both operands expressed in the same coordinate frame. To this end, rotation matrices serve to transform vectors from one coordinate frame to another. Note that the baseline vector $\mathbf{c}$ and the lines-of-sight vectors $\mathbf{v}_1$ and $\mathbf{v}_2$ are all free vectors, so locations of the coordinate frame origins do not matter, and no translations are required. A derivation of the epipolar constraint scalar measurement s starts with the epipolar constraint, which is repeated here as Equation (3.65):

$$\mathbf{v}_1 \cdot (\mathbf{c} \times \mathbf{v}_2) = 0 \tag{3.68}$$

First, replace the baseline vector $\mathbf{c}$ with the equivalent vector expressed in global coordinates contained in Equation (3.64). Also, include a rotation matrix that rotates the vector $\mathbf{c}$

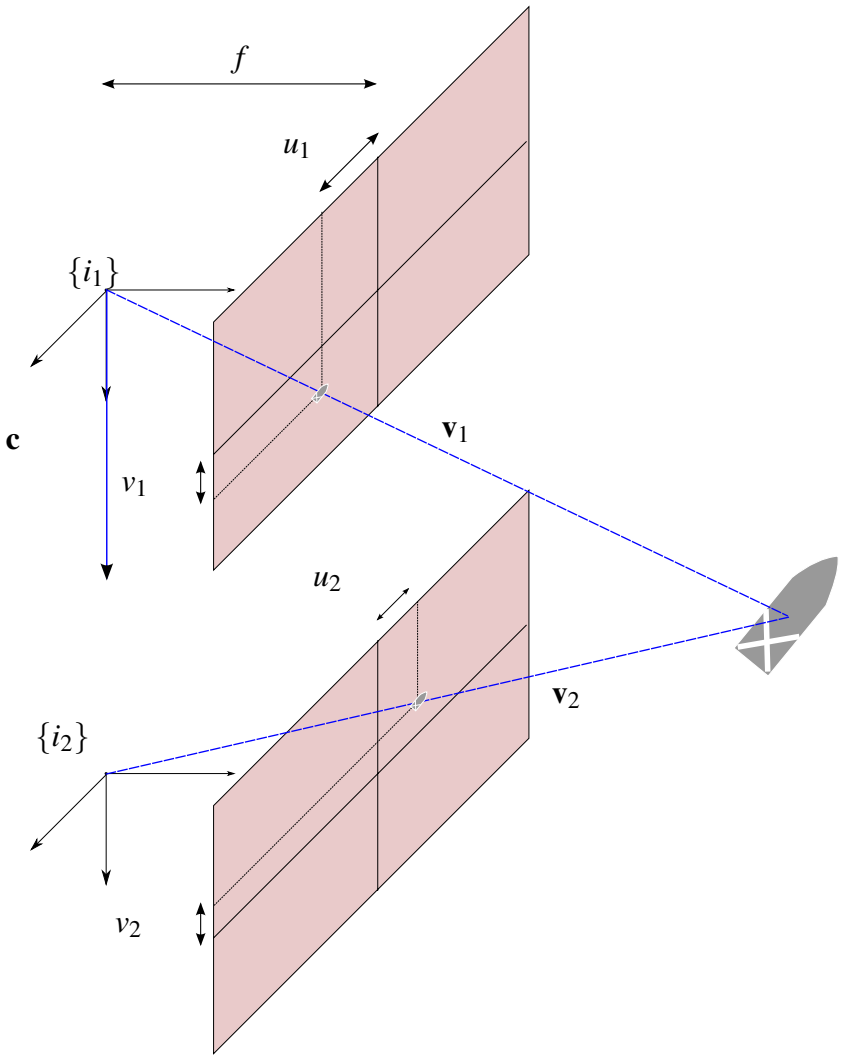


Figure 3.9: Two line-of-sight vectors to the target. The origin of each image plane reference frame has been moved to its respective focal point.

from frame $\{g\}$ to frame $\{i_2\}$ as detailed in Section 3.1.3 and Equation (3.9):

$$\mathbf{v}_1 \cdot \left\{ \mathbf{R}^{\mathrm{T}}(\mathbf{\Lambda}_2) \left({}^{\mathrm{g}}\mathbf{d}_1 - {}^{\mathrm{g}}\mathbf{d}_2 + \Delta^{\mathrm{g}}\mathbf{p}_T \right) \times \mathbf{v}_2 \right\} = 0 \tag{3.69}$$

with the caveat that, at this point, $\mathbf{v}_1$ is still expressed in frame $\{i_1\}$. Next, use the distributive property of the vector cross product to separate the component of the vector $\mathbf{c}$ that is

due to target motion:

$$\mathbf{v}_1 \cdot \left\{ \mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda}_2) \left({}^{\mathrm{g}}\mathbf{d}_1 - {}^{\mathrm{g}}\mathbf{d}_2 \right) \times \mathbf{v}_2 + \mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda}_2) \Delta^{\mathrm{g}}\mathbf{p}_T \times \mathbf{v}_2 \right\} = 0. \tag{3.70}$$

The next step is to move the term associated with the target's motion to the right-hand side, and also reverse the order of the cross product, yielding a positive term:

$$\mathbf{v}_1 \cdot \left\{ \mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda}_2) \left({}^{\mathrm{g}}\mathbf{d}_1 - {}^{\mathrm{g}}\mathbf{d}_2 \right) \times \mathbf{v}_2 \right\} = \mathbf{v}_1 \cdot \left\{ \mathbf{v}_2 \times \mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda}_2) \Delta^{\mathrm{g}}\mathbf{p}_T \right\}. \tag{3.71}$$

For the final step, recall that the cross product within the braces on each side of Equation (3.71) is computed with vectors expressed in frame $\{i_2\}$. For the dot product to be computed, the resultant vector inside the braces must be expressed in frame $\{i_1\}$, which can be accomplished by two successive rotations, first from $\{i_2\}$ to $\{g\}$, then from $\{g\}$ to $\{i_1\}$, yielding:

$$\begin{aligned} \mathbf{v}_1^{\mathrm{T}} \mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda}_1) \mathbf{R}(\boldsymbol{\Lambda}_2) \left\{ \mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda}_2) \left({}^{\mathrm{g}}\mathbf{d}_1 - {}^{\mathrm{g}}\mathbf{d}_2 \right) \times \mathbf{v}_2 \right\} & \\ = \mathbf{v}_1^{\mathrm{T}} \mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda}_1) \mathbf{R}(\boldsymbol{\Lambda}_2) \left\{ \mathbf{v}_2 \times \mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda}_2) \Delta^{\mathrm{g}}\mathbf{p}_T \right\}. & \end{aligned} \tag{3.72}$$

The full expressions for scalar measurement s and predicted scalar value $\hat{s}$ that include these rotation matrices are:

$$s = \mathbf{v}_1^{\mathrm{T}} \mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda}_1) \mathbf{R}(\boldsymbol{\Lambda}_2) \left\{ \mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda}_2) \left({}^{\mathrm{g}}\mathbf{d}_1 - {}^{\mathrm{g}}\mathbf{d}_2 \right) \times \mathbf{v}_2 \right\} \tag{3.73}$$

$$\hat{s} = \mathbf{v}_1^{\mathrm{T}} \mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda}_1) \mathbf{R}(\boldsymbol{\Lambda}_2) \left\{ \mathbf{v}_2 \times \mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda}_2) \Delta^{\mathrm{g}}\mathbf{p}_T \right\}. \tag{3.74}$$

Note that Equation (3.73) contains only the component of the baseline vector due to the observer motion, ${}^{\mathrm{g}}\mathbf{d}_1 - {}^{\mathrm{g}}\mathbf{d}_2$, and Equation (3.74) contains only the components of the baseline vector due to the target motion $\Delta^{\mathrm{g}}\mathbf{p}_T$.

Knowledge of the target's speed V as an element of the state vector simplifies the calculation of predicted scalar value $\hat{s}$. For this method, the target's velocity vector is assumed to be composed of a constant speed V multiplied by a unit vector $\mathbf{u}(\psi)$ in the direction ψ of

the target's course:

$$\Delta^{\mathrm{g}}\mathbf{p}_T = \Delta t \cdot V\mathbf{u}(\psi) \tag{3.75}$$

where Δt is the time elapsed between the two observations. This formulation allows a calculation of predicted scalar value $\hat{s}$ based only on the values of state variables V and ψ and not on two recorded positions. The unit vector $\mathbf{u}(\psi)$ is aligned with the target's course in the global frame. An elementary rotation matrix $\mathbf{R}_t(\psi)$ can rotate a vector from a target-fixed coordinate frame, such as that depicted in Figure 3.10, to the global frame. For example, the rotation matrix:

$$\mathbf{R}_t(\psi) = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{3.76}$$

can rotate a vector representing the target's velocity along course as:

$$\mathbf{R}_t(\psi)\begin{bmatrix} V \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} \cos(\psi) & -\sin(\psi) & 0 \\ \sin(\psi) & \cos(\psi) & 0 \\ 0 & 0 & 1 \end{bmatrix}\begin{bmatrix} V \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} {}^{\mathrm{g}}V_x \\ {}^{\mathrm{g}}V_y \\ 0 \end{bmatrix}. \tag{3.77}$$

where, in this expression, ${}^{\mathrm{g}}V_x$ and ${}^{\mathrm{g}}V_y$ are two components of the target's velocity vector expressed in the global coordinate frame. In this case, the three-element velocity vector on the left-hand side is defined such that the first element represents velocity directly along the target's longitudinal axis. This formulation relies on the assumption that the target's longitudinal axis is exactly aligned with its course. In maritime navigation terms, this assumption is equivalent to assuming that the target has no set and drift due to currents or winds. The developing expression for predicted scalar measurement $\hat{s}$ is, so far:

$$\hat{s} = \mathbf{v}_1^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}(\mathbf{\Lambda}_1)\mathbf{R}(\mathbf{\Lambda}_2)\left\{\mathbf{v}_2 \times \mathbf{R}^{\mathrm{T}}(\mathbf{\Lambda}_2)\mathbf{R}_t(\psi)\Delta t\begin{bmatrix} V \\ 0 \\ 0 \end{bmatrix}\right\}. \tag{3.78}$$

In Equation (3.78), the vector cross-product is actually accomplished by incorporating the elements of vector $\mathbf{v}_2$ into a skew-symmetric matrix $\mathbf{S}(\mathbf{v}_2)$ according to the following ex-

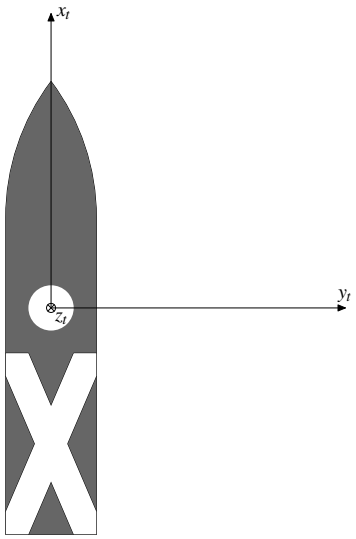


Figure 3.10: Target-centered coordinate reference frame. This reference frame is used to convert the target's velocity vector to the global reference frame.

ample:

$$\mathbf{v} \times \mathbf{c} = \mathbf{S}(\mathbf{v})\mathbf{c} = \begin{bmatrix} 0 & -v_3 & v_2 \\ v_3 & 0 & -v_1 \\ -v_2 & v_1 & 0 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}. \tag{3.79}$$

Furthermore, a matrix multiplication of the state vector produces the velocity vector on the right-hand side of Equation (3.78):

$$\begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ V \\ \psi \end{bmatrix} = \begin{bmatrix} V \\ 0 \\ 0 \end{bmatrix}. \tag{3.80}$$

The modifications in Equations (3.79) and (3.80) incorporated into Equation (3.78) allow predicted scalar value $\hat{s}$ to be defined as the multiplication of row vector $\mathbf{h}^{\mathrm{T}}$ and the target

velocity extracted from estimated state vector $\hat{\mathbf{x}}$:

$$\hat{s} = \underbrace{\mathbf{v}_1^{\mathrm{T}}\mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda}_1)\mathbf{R}(\boldsymbol{\Lambda}_2)\mathbf{S}(\mathbf{v}_2)\mathbf{R}^{\mathrm{T}}(\boldsymbol{\Lambda}_2)}_{\mathbf{h}^{\mathrm{T}}}\mathbf{R}_t(\psi)\begin{bmatrix}0&0&1&0\\0&0&0&0\\0&0&0&0\end{bmatrix}\Delta t\hat{\mathbf{x}} \tag{3.81}$$

where time interval Δt is defined as a multiple of the fundamental sampling rate: $N \cdot T_s$. A compact expression for predicted scalar value $\hat{s}$ follows from Equations (3.77), (3.80), and (3.81):

$$\hat{s} = \mathbf{h}^{\mathrm{T}} N T_s \begin{bmatrix}{}^{\mathrm{g}}V_x\\{}^{\mathrm{g}}V_y\\0\end{bmatrix}, \tag{3.82}$$

recalling that ${}^{\mathrm{g}}V_x$ and ${}^{\mathrm{g}}V_y$ represent two components of the target's velocity vector expressed in the global coordinate frame. Implementation of the full set of estimator equations is easier if the computations can be done such that the measurement vector and measurement matrix have the same dimensions for both in-view and out-of-view modes. Therefore, a predicted scalar measurement $\hat{s}$ is computed using a modified measurement matrix:

$$\hat{s} = \underbrace{\mathbf{h}^{\mathrm{T}} N T_s \mathbf{R}_t(\psi)\begin{bmatrix}0&0&1&0\end{bmatrix}}_{\mathbf{H}'}\hat{\mathbf{x}}. \tag{3.83}$$

For the implementation of this expression in Simulink, the predicted scalar measurement $\hat{s}$ is actually included as the first element of an $m \times 1$ vector with all other elements being zero. In vector form with the extra padding zeros, the predicted pseudomeasurement vector on the left-hand side of Equation (3.83) keeps its dimension $m \times 1$, just as it would be if the regular, in-frame tracking pseudomeasurement were taken. Having the dimension of this vector remain the same for the two modes made the Simulink implementation easier, and made it unnecessary to change the dimension of a signal dynamically in the middle of the simulation. The full summary of all estimator equations is listed in Section 3.3.

In contrast to the conventional Kalman estimation algorithm introduced in Section 3.2.2, this additional measurement using the epipolar constraint uses additional information at the time the target returns to view. The conventional Kalman estimation algorithm does not use this additional information, but instead must set the measurement error covariance

matrix $\mathbf{R}$ to infinity when the target is not in view. This point deserves some special attention. The conventional Kalman estimator, by setting matrix $\mathbf{R}$ to infinity, is ignoring all measurements for the duration that the target is out of view. The conventional estimator is instead using only the most recently-computed position, course, and speed of the target to predict its future position while running the estimation algorithm with no *new* data. This process is also known as *dead-reckoning* the target position.

Once the target returns to view, matrix $\mathbf{R}$ is returned to its nominal value. At this point, the conventional Kalman estimator is once again incorporating measurements, but it is using no information from before the time the target left the field of view.

The estimator using the epipolar measurement is also not using measurements when the target is out of view, because the estimator is extending its sampling interval forward in time until the target returns to view. The algorithm will not make any more computations until that time. When the target returns to view, the error signal in the Kalman state update equation using this epipolar measurement is actually:

$$e = s - \hat{s} = s - \mathbf{h}^{\mathrm{T}} N T_s \begin{bmatrix} {}^{\mathrm{g}}V_x \\ {}^{\mathrm{g}}V_y \\ 0 \end{bmatrix} \tag{3.84}$$

instead of

$$\mathbf{e} = \mathbf{g}(\mathbf{z}) - \mathbf{H}\hat{\mathbf{x}} \tag{3.85}$$

as it would be during normal, target-in-view processing. Another difference between the epipolar estimator and the traditional estimator is that the epipolar estimator uses in its state update equation at the back-in-view instant a multiple of the fundamental sampling interval as defined in Equation (3.81).

The overall benefit of using the epipolar constraint measurement at the return-to-view instant stems from the structure of the row vector $\mathbf{h}^{\mathrm{T}}$. Recall that this row vector is used in the measurement equation for the individual time-step at which the target is first visible having been out of view. Equations (3.67) and (3.81) reviewed together indicate that row vector $\mathbf{h}^{\mathrm{T}}$ has a particular structure that incorporates the epipolar constraint into the measurement equation. According to Kalman estimation theory, this additional information

included in the measurement calculations has the potential to make the overall calculation of the estimate more accurate.

The topic of the epipolar constraint is also mentioned at later points in this chapter. Section 3.3 contains a summary of the Kalman estimation algorithm, including the steps of the epipolar constraint calculation. Furthermore, Section 3.4.2 contains a description of a set of simulations that were designed to evaluate the performance difference between a conventional estimator that uses dead-reckoning (DR), and an estimator that uses the epipolar constraint measurement.

In this case, the number of raw measurements ℓ is 16 because there are two sets of 8 measurements each: one set from the instant of leaving from view, and one set from the instant of returning to view. The dimension m of the transformed raw measurements is unity because the measurement $\hat{s}$ is a scalar; therefore, $\frac{\partial s}{\partial \mathbf{z}}$ has dimension 16×1. In implementation, the 16×1 vector gradient $\frac{\partial s}{\partial \mathbf{z}}$ was calculated by running a symbolic differentiation routine on Equation (3.35) and the resulting expression implemented in the simulation. Also for the simulation, it was convenient for measurement error covariance matrix $\mathbf{R}$ always to have the same dimensions, and not switch from a matrix to a scalar for the intermittent measurements. Therefore, for epipolar measurements, the measurement error covariance matrix $\mathbf{R}$ remained an $m \times m$ matrix; however, it was constructed as an $m \times m$ identity matrix with the upper-left value replaced by the scalar measurement variance value.

3.2.7 Unscented Transform for Measurement Uncertainty

As mentioned in Section 3.2.2, when constructing the measurement equation as in Equation (3.27), while isolating the nonlinearities to the left-hand-side, the consequence is that the elements of the pseudomeasurement noise vector $\boldsymbol{\gamma}$ are no longer normally-distributed random variables. The derivative-based method presented in Section 3.2.2 ignored this potential problem. The potential problem is that the pseudomeasurement noise covariance calculated using Equation (3.35), and based on the covariance of normally distributed variables in measurement noise vector $\mathbf{v}$, may not be accurate if the distribution of elements of $\boldsymbol{\gamma}$ are much different than Gaussian. One technique designed to cope with non-normal distributions is based on the Unscented Transform and its application to estimation is commonly known in the literature as the Unscented Kalman filter (UKF) [99].

To illustrate this point, recall the development of the measurement equation in Section 3.2.2. Combining equations (3.28), (3.29), and (3.30) yields a form of the measurement equation

$$\mathbf{g}(\mathbf{z}^\star + \mathbf{v}) = \mathbf{H}\mathbf{x} + \boldsymbol{\gamma}(\mathbf{z}^\star, \mathbf{v}) \tag{3.86}$$

in which it should be recalled that $\mathbf{z}^\star$ is the perfect, or uncorrupted measurement vector, $\mathbf{v}$ is a vector of assumed zero-mean, normally-distributed measurement noise, and $\boldsymbol{\gamma}$ is the vector of pseudomeasurement noise. After the measurement noise $\mathbf{v}$ is transformed by using the nonlinear transformation $\mathbf{g}$, the resulting pseudomeasurement noise vector $\boldsymbol{\gamma}$ is most likely no longer normally-distributed, yet the autocovariance matrix constructed from $\boldsymbol{\gamma}$ still needs to be computed for the Kalman estimation equations.

The derivative-based approach to this calculation introduced in Section 3.2.2 sought to compute the autocovariance matrix $\mathbf{C}_{\boldsymbol{\gamma}}$ *analytically* using a first-order Taylor series expansion, resulting in Equation (3.35). The UKF relies on a sampling-based approach, explained clearly in Wan and van der Merwe [99], in which a set of sample vectors is transformed by the nonlinear transformation, and the autocovariance matrix is then computed based on that sample set.

For the problem at hand, the setup of the UKF for the computation of the pseudomeasurement autocovariance matrix $\mathbf{C}_{\boldsymbol{\gamma}}$ will be examined first, then the case for the epipolar (returning to view) measurement will be examined. For the measurement equation, let Z be a set of samples chosen to be distributed around a particular noisy raw measurement vector $\mathbf{z}$, then let the result of the transformation of each member $\mathbf{z}_i \in Z$ of this set be:

$$\boldsymbol{\zeta}_i \equiv \mathbf{g}(\mathbf{z}_i). \tag{3.87}$$

Let the random vector $\mathbf{y}$ represent the generalized population of transformed noisy measurement vectors

$$\mathbf{y} = \mathbf{g}(\mathbf{z}) \tag{3.88}$$

as opposed to the specific transformed samples $\boldsymbol{\zeta}_i$. Then, the samples are used to compute

the sample mean $\hat{\mathbf{y}}$ and sample covariance $\widehat{\mathbf{C}}_{\mathbf{y}}$ of the population:

$$\hat{\mathbf{y}} = E\{\mathbf{y}\} = \sum_i W_i \boldsymbol{\zeta}_i \tag{3.89}$$

$$\widehat{\mathbf{C}}_{\mathbf{y}} = \sum_i W_i (\boldsymbol{\zeta}_i - \hat{\mathbf{y}})(\boldsymbol{\zeta}_i - \hat{\mathbf{y}})^{\mathrm{T}}. \tag{3.90}$$

For these equations, each term W_i is a weighting factor designed to ensure that the set Z of raw measurement sample vectors has a probability distribution that has the desired mean and variance. In essence, weights W_i can be seen as probabilities such that, when computing a vector-valued probability mass function using sample values $\mathbf{z}_i \in Z$, each with probability W_i, the mean will be equal to the desired value of the real raw measurement vector $\mathbf{z}$. The probability mass function will also have a diagonal autocovariance matrix with entries equal to the presumed variances of the individual measurements comprising vector $\mathbf{z}$.

The values of the weights depend on dimension ℓ of the raw measurement vector. In the simplest case for the symmetric weighting for the UKF, the sample set Z contains $2\ell + 1$ members: one member is the raw measurement vector actually recorded, which will end up being the sample mean, and 2ℓ other members for which only one element of vector $\mathbf{z}$ is perturbed, both higher and lower, than that element's mean value.

A reduced-order example will illustrate this concept. Note that this example will not include a nonlinear transformation, but merely serves to demonstrate the arrangement of the sample set. Suppose two measurements are available, which are North and East coordinates from GPS. In this case, $\ell = 2$, and the system records (noisy) measurement vector $\mathbf{z}$ with values:

$$\mathbf{z} = \begin{bmatrix} N \\ E \end{bmatrix}. \tag{3.91}$$

A set Z of vectors will be chosen such that the mean value of the set is the recorded vector in Equation (3.91), and the autocovariance of the set of samples is

$$\mathbf{C}_{\mathbf{z}} = \begin{bmatrix} \sigma_N^2 & 0 \\ 0 & \sigma_E^2 \end{bmatrix} \tag{3.92}$$

where the assumption is made that the errors in the North coordinate, and errors in the East coordinate are uncorrelated. The values of the weights are:

$$W_0 = \frac{1}{\ell+1} \qquad \text{and} \qquad W_i = \frac{1}{2\ell+2} \text{ for } 1 \leq i \leq 2\ell \tag{3.93}$$

The values of the $2\ell+1 = 5$ members of the sample set are constructed from recorded vector $\mathbf{z}$ such that:

$$\mathbf{z}_0 = \mathbf{z} \qquad \text{and} \qquad \mathbf{z}_i = \begin{cases} \mathbf{z} + \sqrt{\ell+1}\sigma & \text{for } 1 \leq i \leq \ell \\ \mathbf{z} - \sqrt{\ell+1}\sigma & \text{for } \ell < i \leq 2\ell \end{cases} \tag{3.94}$$

where the value σ is the variance for that element of the raw measurement vector. Thus, for this example:

$$Z = \left\{ \mathbf{z}_0 = \begin{bmatrix} N \\ E \end{bmatrix}, \mathbf{z}_1 = \begin{bmatrix} N + \sqrt{3}\sigma_N \\ E \end{bmatrix}, \mathbf{z}_2 = \begin{bmatrix} N \\ E + \sqrt{3}\sigma_E \end{bmatrix}, \right. \\ \left. \mathbf{z}_3 = \begin{bmatrix} N - \sqrt{3}\sigma_N \\ E \end{bmatrix}, \mathbf{z}_4 = \begin{bmatrix} N \\ E - \sqrt{3}\sigma_E \end{bmatrix} \right\}. \tag{3.95}$$

It can be verified that with weights according to Equation (3.93)

$$W_0 = \frac{1}{3} \qquad \text{and} \qquad W_1 = W_2 = W_3 = W_4 = \frac{1}{6} \tag{3.96}$$

then the sample mean $\hat{\mathbf{z}}$ and sample autocovariance matrix $\widehat{\mathbf{C}}_{\mathbf{z}}$ are

$$\hat{\mathbf{z}} = \sum_{j=0}^{2\ell} W_j \mathbf{z}_j = \begin{bmatrix} N \\ E \end{bmatrix} \tag{3.97}$$

$$\widehat{\mathbf{C}}_{\mathbf{z}} = \sum_{j=0}^{2\ell} W_j (\mathbf{z}_j - \hat{\mathbf{z}})(\mathbf{z}_j - \hat{\mathbf{z}})^{\mathrm{T}} = \begin{bmatrix} \sigma_N^2 & 0 \\ 0 & \sigma_E^2 \end{bmatrix}. \tag{3.98}$$

Equations (3.97) and (3.98) indicate that the set of samples was constructed to have a sample mean and sample covariance consistent with *a priori* knowledge of the noisy measurement vector.

For the full estimation problem in the case in which the target is in the field of view, the dimension of the measurement vector $\ell = 10$; in other words, $\mathbf{z} \in \mathbb{R}^{10\times 1}$, as mentioned in Section 3.2.4. The nonlinear transformation $\mathbf{g}$ comprises both the matrix $\mathbf{M}^{-1}$ of Equation (3.17), and also the matrix $\mathbf{S}^{-1}$ of Equation (3.58). The result of nonlinear transformation $\mathbf{g}(\mathbf{z})$ is a 3×1 vector composed of two coordinates of target position in the global coordinate frame, and the angular measure of the target's course also in the global coordinate frame.

The transformed samples are then

$$\boldsymbol{\zeta}_i = \mathbf{g}(\mathbf{z}_i) \qquad \text{for } 0 \leq i \leq 2\ell \tag{3.99}$$

and the sample mean value of the resultant set of 3×1 vectors is

$$\hat{\mathbf{y}} = \sum_{j=0}^{2\ell} W_j \boldsymbol{\zeta}_j. \tag{3.100}$$

The autocovariance matrix has dimension 3×3, and it is computed by

$$\widehat{\mathbf{C}}_{\mathbf{y}} = \sum_{j=0}^{2\ell} W_j (\boldsymbol{\zeta}_j - \hat{\mathbf{y}})(\boldsymbol{\zeta}_j - \hat{\mathbf{y}})^{\mathrm{T}}. \tag{3.101}$$

Thus, the 3×3 matrix $\widehat{\mathbf{C}}_{\mathbf{y}}$ is used in the Kalman estimation equations for measurement error covariance matrix $\mathbf{R}$ in the equation for the calculation of Kalman gain, and the 3×1 vector $\hat{\mathbf{y}}$ is used as the vector of measurements in the equation that contains the state estimate update. It may seem unusual that the mean vector of the transformed samples $\hat{\mathbf{y}}$ is used in the Kalman estimation equations in place of the transformed version of the actual measurement vector $\mathbf{z}$ that was recorded. In theory, the mean of the transformed samples $\hat{\mathbf{y}}$, is more likely to represent the true mean of the transformed measurement vector than the recorded measurement vector $\mathbf{z}$, even though the set of samples was constructed to have $\mathbf{z}$ as their mean.

For the full estimation problem in the case in which the target has just returned to the field of view, and the epipolar measurement is used, the dimension ℓ of the measurement vector

is now 20 because there is a set of 10 values for the last in-view instant, and another 10 values for the first return-to-view instant. Slightly different symbols will be used in this case in order to distinguish this case from that just mentioned for the previous in-view measurement. For this case, the nonlinear transformation **g** that will be used is embodied in Equation (3.73), in which a set of 20 raw measurements are transformed into one scalar measurement $\hat{s}$. The transformed samples used to calculate s are:

$$\boldsymbol{\varsigma}_i = \mathbf{g}(\mathbf{z}_i) \qquad \text{for } 0 \leq i \leq 2\ell. \tag{3.102}$$

The transformed versions of the $2\ell + 1 = 41$ samples have the symbol ς_i and these are all scalar values. The symbol ς is a variation of the Greek letter sigma known as the lunate sigma, and is slightly different than the Greek letter zeta, ζ, used in Equation (3.99). The mean and variance of the transformed samples are given by:

$$\hat{y} = \sum_{j=0}^{2\ell} W_j \varsigma_j \tag{3.103}$$

$$\widehat{C}_y = \sum_{j=0}^{2\ell} W_j (\varsigma_j - \hat{y})(\varsigma_j - \hat{y})^{\mathrm{T}}. \tag{3.104}$$

Thus, the scalar value $\hat{y}$ is used as the value for measurement $\hat{s}$, and scalar value $\widehat{C}$ is incorporated into a measurement error covariance matrix **R**. As in the in-view case, the value actually used in the Kalman estimation equations the mean of the transformed samples $\hat{y}$, and not the value of s that results from performing the nonlinear transformation on the recorded measurement vector **z**.

3.3 Summary of the Dual-rate Estimation Algorithm

As the descending ADS uses its visual estimation algorithm in flight to compute the target's position, course, and speed, much of the challenge in doing these computations stems from the swinging motion of the parafoil that causes the target to be out of view for significant durations of time. When the target is out of view, there are no measurements for the visual estimation algorithm to use in its computations. A traditional Kalman estimator in this case would set its measurement error covariance matrix to have extremely high values, allowing

the estimation algorithm to ignore missing measurements and to compute estimates based only on the underlying state-space model. As explained in Section 3.2.6, the traditional Kalman estimator is therefore computing only the DR target position when the target is out of view.

The algorithm described in this chapter follows a different approach. This algorithm has two different models, or modes: one mode for the target in view, and one for the target out of view. A traditional Kalman estimator is likely to maintain the same sampling rate whether the target is in view, or the target is out of view and measurements are being ignored. For the algorithm described in this chapter, the mode for which the target is out of view has a variable sampling rate. The out-of-view mode takes one sample at the instant that the target is leaving the image frame, and another sample at the instant that the target returns to the image frame. For the time interval between these two instants, the estimator is neither taking samples nor performing computations. Because these intervals are variable, the estimator in this mode is considered to have a variable sampling rate, different from the constant rate of the in-frame mode. For this reason, the estimator described in this chapter is a dual-rate estimator. Both modes, or models, are represented below:

$$\begin{cases} \mathbf{x}[n+1] = \boldsymbol{\Phi}(T_s)\mathbf{x}[n] + \mathbf{G}\mathbf{w}[n] & \text{in view} \\ \mathbf{x}[n+1] = \boldsymbol{\Phi}(NT_s)\mathbf{x}[n] + \mathbf{G}\mathbf{w}[n] & \text{returning to view} \\ \mathbf{g}(\mathbf{z}[n]) = \mathbf{H}\mathbf{x}[n] + \boldsymbol{\gamma}[n] & \text{in view} \\ s[n] = \mathbf{h}^{\mathrm{T}} NT_s \begin{bmatrix} {}^{\mathrm{g}}V_x \\ {}^{\mathrm{g}}V_y \\ 0 \end{bmatrix} + \boldsymbol{\gamma}[n] & \text{returning to view} \end{cases} \tag{3.105}$$

Note that the state equations were written in terms of discrete-time state transition matrix $\boldsymbol{\Phi}$, even though in practice, the state update is calculated using the nonlinear state update function $\mathbf{f}$. Furthermore, the state transition matrix is written with the sampling interval as an explicit argument so that the two rates of the two modes are clearly shown.

In this dual-mode setup, the estimator *does not operate* during the interval that the target is out of view; therefore, the state vector at the returning-to-view instant would be $\mathbf{x}[n+1]$ if at the out-of-view instant it were $\mathbf{x}[n]$. For the in-view mode, processing is done in the

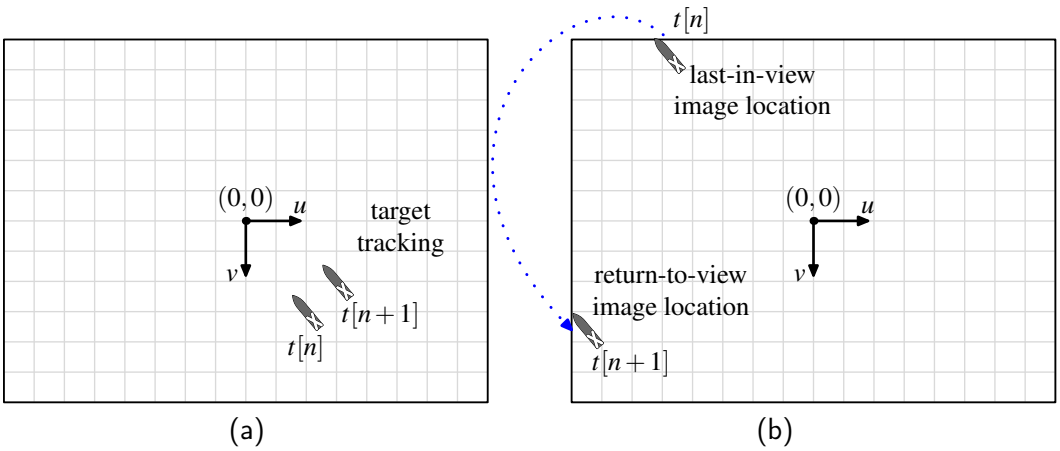


Figure 3.11: Two estimation modes. Target images on the image plane are shown for the in-view tracking mode (a) and the returning-to-view mode (b).

conventional manner: times n and $n+1$ represent sequential processing cycles, each within duration T_s. Figure 3.11 contains a representation of the target image on the image plane for both the in-view and returning-to-view modes along with notations for the target image locations at times $t[n]$ and $t[n+1]$.

The portion of the estimation algorithm that handles the constant-sampling-rate in-frame view condition is straightforward and has been detailed in Sections 3.2.1 and 3.2.2. This section describes the implementation of the discrete-time estimation algorithm for the out-of-frame mode and provides details regarding computation of scalar measurement $\hat{s}$ using sampling-based methods (the Unscented Transform). The convention for the notation below is that *a priori* values, which are those made before the latest measurement has been incorporated, are denoted by a superscript $-$ symbol, and the *a posteriori* values, those calculated after incorporating the latest measurement, are denoted by a superscript $+$ symbol. The complete algorithm is listed below; and, in particular, the discussion of the computations of the measurement error covariance matrix $\mathbf{R}$, and the *a posteriori* state vector estimate, $\mathbf{x}^+$, should further clarify the material in Sections 3.2.6 and 3.2.7. A list of the variables used in the algorithm is included in the nomenclature list on page xix. Notice that in that list, a symbol can have more than one meaning. For a particular usage of a symbol, the desired meaning should be apparent from the context.

1. Calculate the *a priori* error covariance matrix:

$$\mathbf{P}^{-}[n] = \mathbf{\Phi}[n-1]\mathbf{P}^{+}[n-1]\mathbf{\Phi}^{\mathrm{T}}[n-1] + \mathbf{Q}[n-1]. \tag{3.106}$$

   The discrete-time state transition matrix is only used in this step, and is computed using the matrix exponential:

$$\mathbf{\Phi} = e^{\mathbf{F}T_s}. \tag{3.107}$$

2. Calculate the *a priori* state estimate for the alternate state formulation in Section 3.2.4 using nonlinear function **f**:

$$\begin{cases} \hat{\mathbf{x}}^{-}[n] = \hat{\mathbf{x}}^{+}[n-1] + \mathbf{f}(\hat{\mathbf{x}}^{+}[n-1])T_s & \text{in view} \\ \hat{\mathbf{x}}^{-}[n] = \hat{\mathbf{x}}^{+}[n-1] + \mathbf{f}(\hat{\mathbf{x}}^{+}[n-1])NT_s & \text{returning to view} \end{cases} \tag{3.108}$$

   The method used to compute the *a priori* state estimate in Equation (3.108) is one-step Euler integration. The difference in the computation between the in-view target tracking mode and the return-to-view epipolar mode is that for the epipolar mode, the sampling interval is extended as a multiple N of the fundamental sampling interval T_s. Also note that an additional difference between the two modes is that, when considering the state equation (see the first two lines of Equation (3.105)), for the return-to-view mode, the process noise covariance matrix $\mathbf{Q}$ must be computed taking into account the longer sampling interval NT_s. This longer sampling interval could be used in Equation (3.24) to compute the return-to-view process noise covariance matrix $\mathbf{Q}$; however, in practice, the longer sampling interval is used with the Van Loan algorithm for this computation as described in Section 3.2.1.
3. Calculate the Kalman gain. As a prerequisite, measurement error covariance matrix $\mathbf{R}$ is calculated using sampling-based (UKF, Section 3.2.7) methods, according to the following procedure.

   (a) Select sigma points for the Unscented Transform. If the target is in view, then measurement vector $\mathbf{z}$ has length $\ell = 10$, and set of sigma points Z has $2\ell + 1 = 21$ members. If the target is returning to view, then measurement vector $\mathbf{z}$ has length $\ell = 20$, and set of sigma points Z has $2\ell + 1 = 41$ members.

   (b) Compute the transformed samples using the individual members $\mathbf{z}$ of set Z: If the target is in view, then the transformed samples have the symbol ζ. If the

target is returning to view, then the transformed samples have the symbol ς.

$$\begin{cases} \boldsymbol{\zeta}_i = \mathbf{g}(\mathbf{z}_i) & \mathbf{g}(\cdot) : \mathbb{R}^{10\times 1} \to \mathbb{R}^{3\times 1} \quad \text{for } 0 \le i \le 2\ell \quad \text{target in view} \\ \varsigma_i = \mathbf{g}(\mathbf{z}_i) & \mathbf{g}(\cdot) : \mathbb{R}^{20\times 1} \to \mathbb{R} \quad\quad\quad \text{for } 0 \le i \le 2\ell \quad \text{target returning to view} \end{cases} \tag{3.109}$$

(c) Compute the mean of transformed samples (vector or scalar) represented by random variable Y using weighting values W.

$$\begin{cases} \hat{\mathbf{y}} = \displaystyle\sum_{j=0}^{2\ell} W_j \boldsymbol{\zeta}_j & \text{target in view} \\ \hat{y} = \displaystyle\sum_{j=0}^{2\ell} W_j \varsigma_j & \text{target returning to view} \end{cases} \tag{3.110}$$

(d) Compute the covariance of transformed samples (vector or scalar) represented by random variable Y using weighting values W.

$$\begin{cases} \widehat{\mathbf{C}}_{\mathbf{y}} = \displaystyle\sum_{j=0}^{2\ell} W_j (\boldsymbol{\zeta}_j - \hat{\mathbf{y}})(\boldsymbol{\zeta}_j - \hat{\mathbf{y}})^{\mathrm{T}} & \text{target in view} \\ \widehat{C}_y = \displaystyle\sum_{j=0}^{2\ell} W_j (\varsigma_j - \hat{y})(\varsigma_j - \hat{y})^{\mathrm{T}} & \text{target returning to view} \end{cases} \tag{3.111}$$

(e) Construct measurement error covariance matrix $\mathbf{R} \in \mathbb{R}^{m\times m}$ based on covariance of transformed samples. Recall that for the returning-to-view case, the measurement is a scalar, $\hat{s}$. For implementation in Simulink, though, value $\hat{s}$ is inserted as the first entry in an $m \times 1$ vector with the other entries being zero. As a consequence, the covariance value for the scalar measurement is inserted as the top-left entry of an $m \times m$ identity matrix so that the matrix dimensions are the same for both the in-view and the returning-to-view cases.

$$\mathbf{R}[n] = \begin{cases} \widehat{\mathbf{C}}_{\mathbf{y}} & \text{target in view} \\ \begin{bmatrix} \widehat{C}_y & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} & \text{target returning to view} \end{cases} \tag{3.112}$$

(f) Use measurement error covariance matrix $\mathbf{R}$ to compute gain matrix $\mathbf{K}$. For the case of using scalar measurement with target returning to view, replace measurement matrix $\mathbf{H}$ with modified measurement matrix $\mathbf{H}'$ defined in Equation (3.83).

$$\mathbf{K}[n] = \begin{cases} \mathbf{P}^{-}[n]\mathbf{H}^{\mathrm{T}}[n]\left[\mathbf{H}[n]\mathbf{P}^{-}[n]\mathbf{H}^{\mathrm{T}}[n] + \mathbf{R}[n]\right]^{-1} & \text{target in view} \\ \mathbf{P}^{-}[n]\mathbf{H}'^{\mathrm{T}}[n]\left[\mathbf{H}'[n]\mathbf{P}^{-}[n]\mathbf{H}'^{\mathrm{T}}[n] + \mathbf{R}[n]\right]^{-1} & \text{target returning to view} \end{cases} \tag{3.113}$$

4. Calculate the *a posteriori* state vector estimate using the computed sample mean ($\hat{\mathbf{y}}$ or $\hat{y}$). For the case of the returning-to-view measurement, replace the measurement matrix $\mathbf{H}$ with modified measurement matrix $\mathbf{H}'$ defined in Equation (3.83), recalling that the product of $\mathbf{H}'$ and any vector results in a vector in which all but the first element are zero. This was done so that the dimension $n \times m$ of gain matrix $\mathbf{K}$ in step 3f would remain the same for both estimation modes. Therefore, scalar measurement $\hat{s}$ is the first and only non-zero element in an $m \times 1$ vector. The mean of the transformed samples, $\hat{\mathbf{y}}$ or $\hat{y}$, takes the place of the actual measurement, as described in Section 3.2.7.

$$\hat{\mathbf{x}}^{+}[n] = \begin{cases} \hat{\mathbf{x}}^{-}[n] + \mathbf{K}[n]\left[\hat{\mathbf{y}}[n] - \mathbf{H}[n]\hat{\mathbf{x}}^{-}[n]\right] & \text{target in view} \\ \hat{\mathbf{x}}^{-}[n] + \mathbf{K}[n]\left(\begin{bmatrix} \hat{y}[n] \\ 0 \\ 0 \end{bmatrix} - \mathbf{H}'[n]\hat{\mathbf{x}}^{-}[n]\right) & \text{target returning to view} \end{cases} \tag{3.114}$$

5. Calculate the *a posteriori* error covariance matrix. For the case of using scalar measurement with target returning to view, replace measurement matrix $\mathbf{H}$ with modified measurement matrix $\mathbf{H}'$ defined in Equation (3.83).

$$\mathbf{P}^{+}[n] = \begin{cases} [I - \mathbf{K}[n]\mathbf{H}[n]]\,\mathbf{P}^{-}[n] & \text{target in view} \\ [I - \mathbf{K}[n]\mathbf{H}'[n]]\,\mathbf{P}^{-}[n] & \text{target returning to view} \end{cases} \tag{3.115}$$

6. Compute state transition matrix using Equation (3.107); then, compute process noise covariance matrix.

$$\mathbf{Q}[n] = \int_{0}^{T_s} \mathbf{\Phi}(\tau)\mathbf{Q}_c\mathbf{\Phi}^{\mathrm{T}}(\tau)\, d\tau \tag{3.116}$$

3.4 Simulation of the Kalman Estimator

Implementing the estimation algorithm described in Sections 3.2 and 3.3 in a simulation environment such as MATLAB and Simulink facilitated checking the algorithm for proper operation before any assessment of the algorithm's performance was done. Furthermore, implementation of the algorithm in a programming language such as MATLAB is a prerequisite to writing code in the C programming language for the microcontroller on the autopilot itself.

The goal of the simulation effort of this portion of the research was to produce a numerical model of the Kalman estimator that could ultimately be tested with recorded experimental data. A necessary first step was proving that correct results could be achieved in a simulation that included noise-free models of the ADS and the moving surface target. This initial simulation also included a process model with the capability to calculate correctly the relative position between ADS and target in the image frame, and to compute the associated image plane measurements available to the ADS. Once this model was proven, then the next step was to replace the truth model for the ADS in the simulation with one containing look-up tables that implemented the actual recorded data from a flight test experiment conducted at Camp Roberts, California, that included a moving ground target.

3.4.1 Estimation Algorithm Verification Model

The first simulation that was developed in Simulink included very simple artificial models of the observer's true trajectory and of the surface target's true trajectory which facilitated checking the correctness of the estimation algorithms. In this simulation, the perspective-projective model was implemented that computed the true values of the image plane measurements (u, v, ψ_i) available to the observer throughout the flight. The perspective-projective model as described in Section 3.1.2 was used to calculate image plane measurements in units of actual length of image features on the image plane. In this simulation, the image plane measurements were converted from units of length to units of number of *picture elements* (pixels) based on a representative image size of 640 columns of pixels, and 480 rows of pixels, known as 640×480. The relationship between length and numbers of pixels was established using the representative image plane size of the small digital camera used in the flight tests as described in Appendix B. The image plane measurements (u, v, ψ_i) were then converted from units of number of pixels back to units of

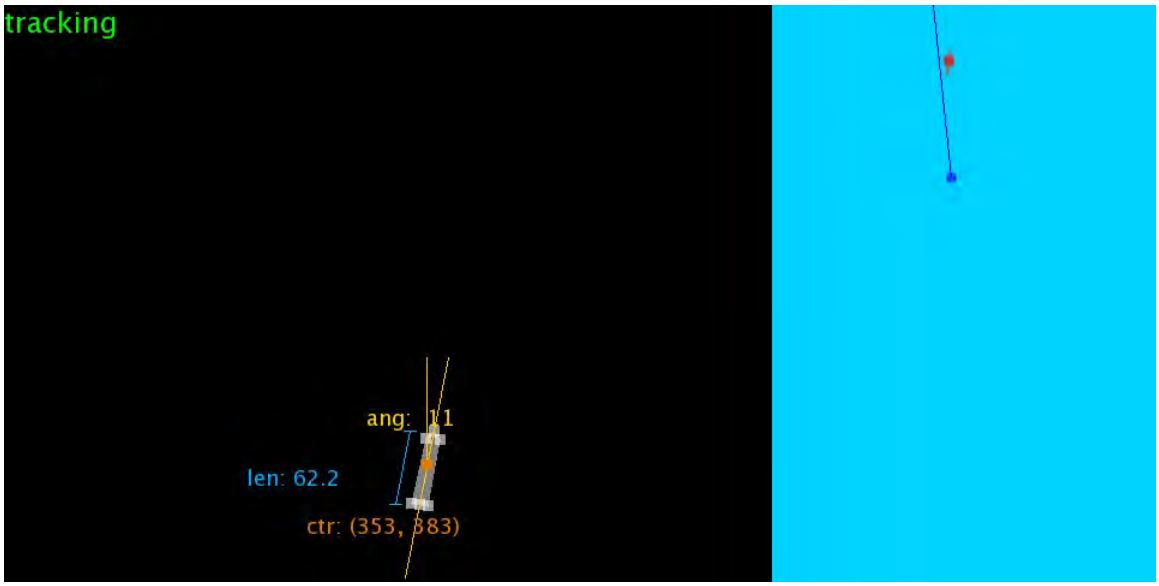


Figure 3.12: Artificial trajectory simulation image plane and overhead. The image plane is on the left, and the overhead depiction of the scene is on the right.

length before they were passed to the estimation algorithm. Therefore, some quantization noise was introduced at this stage; however, no attempt was made to introduce any other sources of measurement noise in this simulation because the purpose of this first effort was only the validation of this simplest of models.

The geometry of this simple scenario is depicted in Figure 3.12. An instant with the target in view is shown with the image plane view on the left, and an overhead depiction of the scene on the right. The observer and target are both moving in simple, straight-line trajectories, with the observer approaching the target from astern. Furthermore, the target's motion includes oscillations in yaw only so that the target's image periodically swings out of view.

3.4.2 Epipolar Versus DR Comparison Model

The second version of this simple simulation was designed to evaluate the benefit of the epipolar constraint measurement over the method of simply using the target's DR position when it is not in view. A very convenient artificial motion was programmed for the observer in this simulation to achieve a wide separation between the last-in-view and return-to-view focal point positions. In Figure 3.13, the observer is moving North to South and crossing the

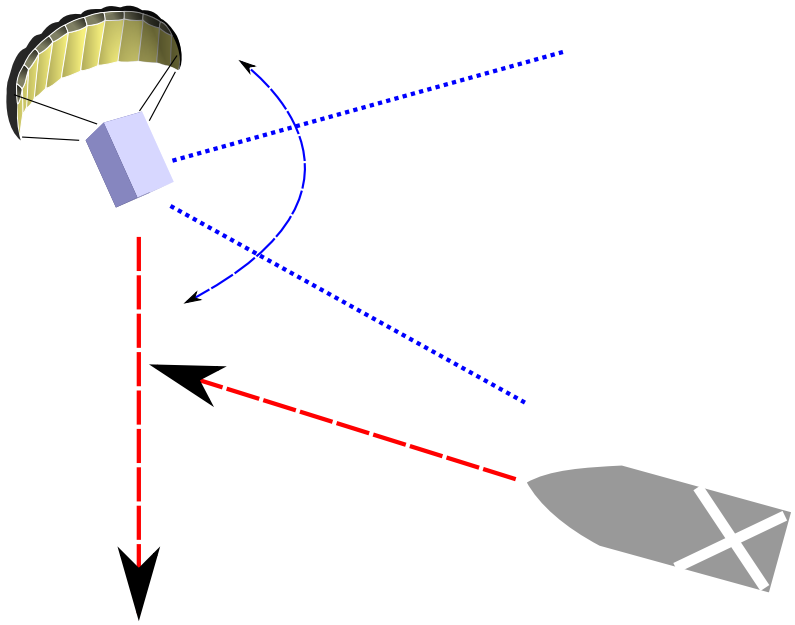


Figure 3.13: Geometry of simulation to evaluate epipolar measurement. The observer moves across the target's course so that last-in-view and return-to-view locations are widely separated.

target's course while its field of regard rotates in the horizontal plane. For this simulation, the observer was programmed to turn away from the target after a 3 s period of observation, then turn back toward the target after a time delay of 130 s, resulting in approximately two minutes during which the target was out of view. This setup enabled a much larger out-of-frame duration than was possible in the flight-test-based hybrid simulation described in Section 3.4.4.

Beginning with this simulation and continuing through subsequent versions, there were four different Kalman estimation algorithms executed separately on the data:

linear estimator uses the simple state vector described in Section 3.2.1, and it includes the epipolar calculation;

extended with DR uses the alternate state vector described in Section 3.2.4, and it uses the DR method when the target is out-of-frame;

extended with epipolar also uses the alternate state vector described in Section 3.2.4 and it uses the epipolar constraint measurement described in Section 3.2.6 for the back-in-frame instant;

Unscented estimator algorithm is post-processed in MATLAB rather than included in the Simulink model; it uses the sampling-based method of computing measurement error detailed in Section 3.2.7.

The parameters that were adjusted in this model were measurement noise and measurement error covariance matrix **R**. For the first two sets of simulations, actual added process noise was kept at zero to discern the effect of the two varied parameters. The objective of this set of simulations was to determine whether the use of the epipolar measurement at the back-in-frame instant conferred any advantage over an estimator that simply ignored missing measurements when the target was out of view, and resumed normal processing when the target returned to view—the DR method. When process noise and measurement noise are zero, the dead reckoning estimator would likely produce very accurate results; therefore the three epipolar and one DR estimator mentioned above were compared under several conditions of noise, and *a-priori* knowledge of noise (matrix **R**).

3.4.3 Demonstration Hybrid Model Using Flight Test Data

The simulations described in Section 3.4.1 and Section 3.4.2 contained very simple models of observer motion that included translation and rotation about the observer's yaw axis only. To represent more realistic observer motion that included roll, pitch, and yaw, the next simulation model calculated the truth data for the observer's trajectory using a portion of the recorded motion data from Snowflake's autopilot from a flight test at Camp Roberts, California in May 2011. Additional details about the setup of this experiment with an actual moving target are contained in Appendix B. The portion of the flight test that was modeled in the simulation is approximately 1.2 s near the end of the flight as Snowflake approached the target vehicle on a curving flight path and the vehicle was visible in the recorded video from a camera attached to Snowflake's external casing. This simulation is therefore a combination of flight test data and simulation; flight test data is used to create the truth data for the simulation. The limitation in this case is that the portion of the recorded flight during which there was a visible target whose location was know was very short in duration.

The overhead view of observer and target in this simulation and the associated image plane view are shown in the bottom two panels of Figure 3.14. In addition, the top two panels

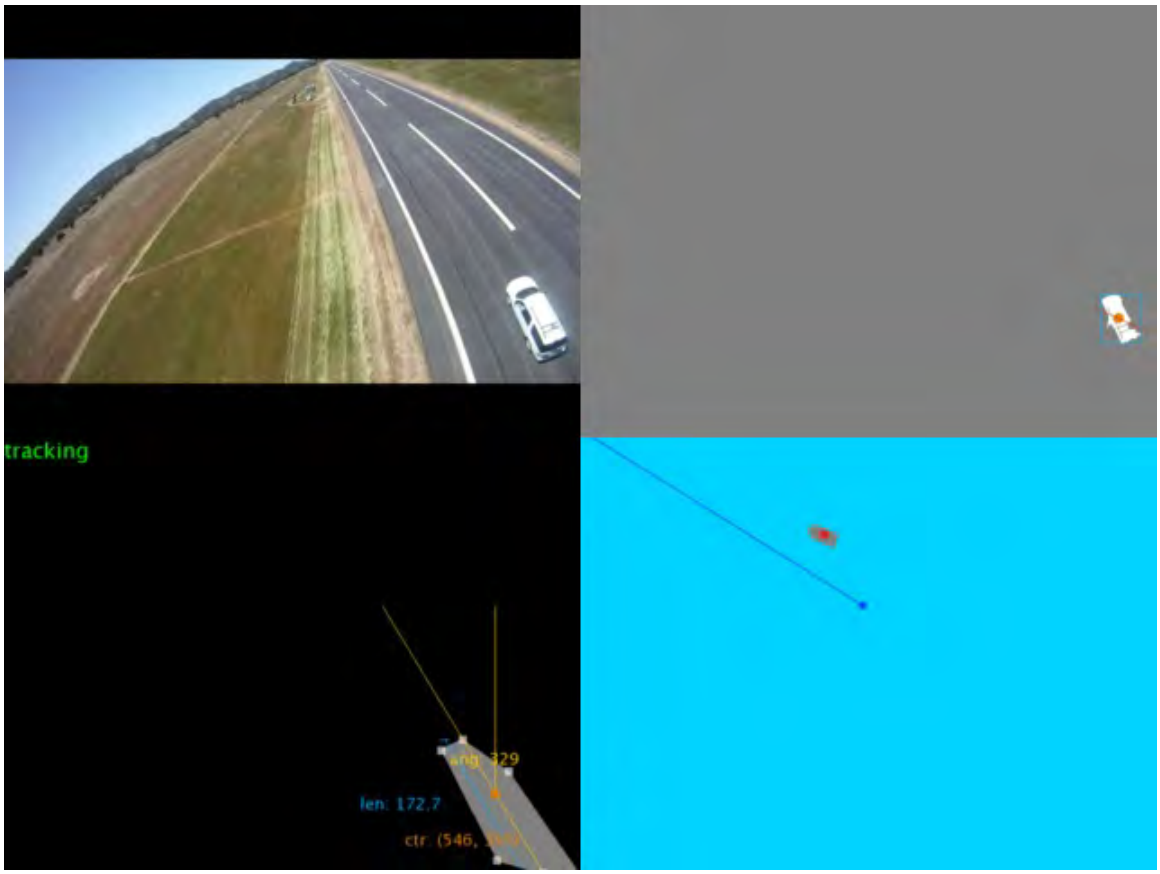


Figure 3.14: Hybrid demonstration simulation raw and processed video. The raw flight video is in the upper-left panel, processed flight video is in the upper-right, the simulation image plane is in the lower-left, and the simulation overhead view is in the lower-right panel.

of Figure 3.14 show raw and processed video that was collected during the flight test. The upper-left panel contains a frame of the original raw video collected from the flight test, and the upper-right panel contains a frame of video that has been processed in order to extract the segment of the video frame containing the target and also to compute the location of the centroid of the target image. For this simulation, the simulation parameters, such as the starting relationship between observer and target, were adjusted so that the simulation output (image plane) would resemble frames of the raw video. In this way, the simulation was made to match, in an approximate manner, observer and target trajectories from the flight test.

This first set of flight test data and simulation was not designed to produce analytical results

due to the very short duration of time that the target appeared in frame, and the fact that the target did not exit and re-enter the field of view for a long enough period of time to provide a good evaluation of target estimator performance. Rather, this simulation was designed to be an end-to-end demonstration showing the input video (upper-left), video processing (upper-right), allowing target measurements (lower-left), leading to estimation of the target (lower-right).

3.4.4 Trade Study of the Hybrid Model

The aforementioned hybrid of flight test data and simulation notwithstanding, a simulation was needed that actually did allow for comprehensive analysis of the Kalman estimator. The Snowflake ADS has had quite an active program of flight testing since its initial development in 2008 [11], and the work described in this dissertation has relied heavily on these tests, the pertinent details of which are contained in Appendix B. Therefore, it was a goal that for the analysis of a visual sensing system for terminal guidance that, in the absence of an actual operational prototype visual sensing system, the necessary simulations be made as realistic as possible using flight test data. With this goal in mind, the hybrid simulation introduced in this section was revised to model a geometry between the observer and target that would enable meaningful analysis of estimator performance.

An example of the video output of this simulation is shown in Figure 3.15. This video display was used as a check that the simulation was functioning properly; however, the real quantitative output of this simulation error was estimation error versus time for various estimator configurations. The results that will be described from this simulation are derived from a trade study analysis of these different estimator configurations. The different simulation runs were conducted with the same random inputs (random number generator seeds were not changed) and same flight geometry and trajectory so that a proper comparison could be made of the estimator performance in the different cases. Three different sets of estimator parameters used in this model are summarized in Table 3.4.

The nominal parameter set represented the current characteristics of the Snowflake ADS using the UAH Advanced Autopilot System (AAS), which executes the main loop of the software at a 4 Hz rate. While there is no integrated visual sensor currently in the Snowflake hardware, the nominal image plane size was chosen to conform to the Video Graphics

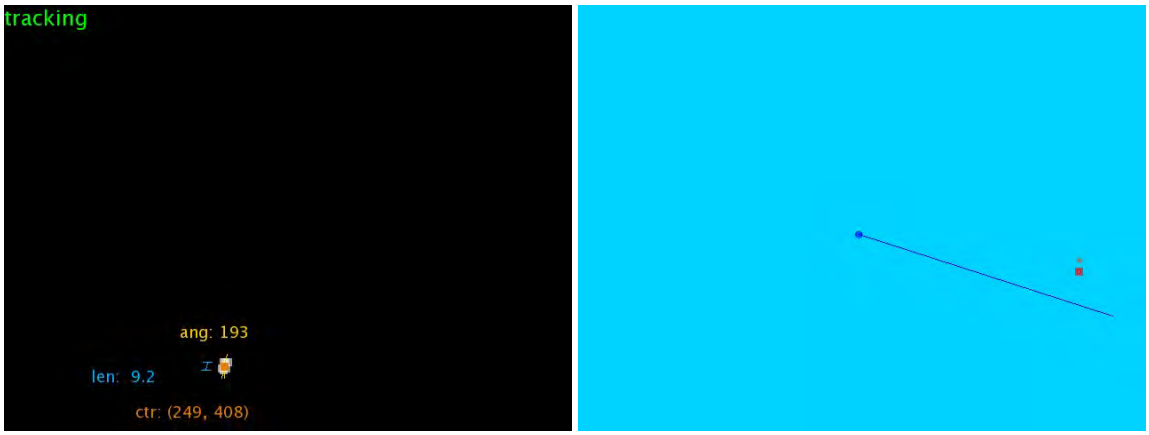


Figure 3.15: Hybrid simulation used for estimator performance analysis. The simulation image plane is on the left and the overhead view is on the right.

Table 3.4: Nominal and improved sets of estimator parameters. The hybrid simulation for performance analysis of the estimator calculated estimation errors using each of these parameter sets.

parameter set	estimator sampling rate	image plane size/pixels
nominal	4 Hz	640×480
improved	20 Hz	1600×1200
fast	20 Hz	640×480

Array (VGA) display standard.

The improved parameters represent hardware and software modifications that could be achieved in the near term. An overall estimator sampling rate of 20 Hz would require an upgrade to the autopilot's GPS receiver, whereas the autopilot's angular rate gyroscopes and the visual sensor's frame rate both already have the capability to exceed 20 Hz. The fast parameter set represents a compromise in which the estimator sampling rate in increased, but the image plane resolution remains at the nominal value.

Even more important than the characteristics listed in Table 3.4 is the quantitative value of measurement noise. Measurement noise was applied to the numerical data representing GPS coordinates and autopilot angular orientation measurements. Error in GPS measurements was addressed in Section 3.2.3, and a very basic measurement noise model was developed there. In contrast to that simple error model, a more in-depth text on navigation

systems, such as Rogers [100], explains that GPS errors are due to time-varying biases and drift rates in the user clock. The more sophisticated error model presented in Rogers would produce estimates of position error that are quite different from those produced by a simple additive white noise model; however, the simple model was chosen to make implementation in the simulation easier. In the following, an alternative model of measurement error is developed that relies on modeling the GPS measurement error and also on-board angular orientation measurements using actual recorded flight test data.

A well-planned experiment enables comparison of measured values with known *true* values. Unfortunately, in the flight tests described in Appendix B, there was no *external* observation system that could provide more precise position information against which on-board GPS measurements could be compared. In order to create the more realistic noise model, the recorded flight test telemetry had to be used as a basis for both the truth and also the measured data sets. Using data from one flight test conducted on 2 May 2011, the details of which are contained in Appendix B, a long, straight segment of the flight was chosen that contained several instances of the target moving in and out of the observer's field of view for durations long enough to make meaningful assessments of estimator performance. The segment of the flight chosen for this simulation is shown in Figure 3.16, which corresponds to phase 4 and phase 5 of the Snowflake ADS guidance algorithm.

In order to create the model for the true observer positions, the observer trajectory was analyzed in terms of measured GPS coordinate values and measured angles, and a piece-wise cubic spline was calculated to fit a subset of the data points from the telemetry series. The spline was only fit to a subset because it was thought that, for example, a spline fit to every seventh point would have the intervening six points distributed around it, some above, some below. Figure 3.17 contains example segments of recorded flight data for the GPS north coordinate and for the accelerometer sensor ϕ angle measurement with additional plots of the subset of points used to calculate the spline fit, and the spline fit itself. All three measured GPS coordinates and all three measured orientation angles were treated in this fashion to calculate six piece-wise cubic spline polynomials that were used to represent the true position of the observer.

The GPS data shown in Figure 3.17 is already very smooth, so that the recorded data points are all close to the spline fit; however, the data for angle ϕ has more varied values above

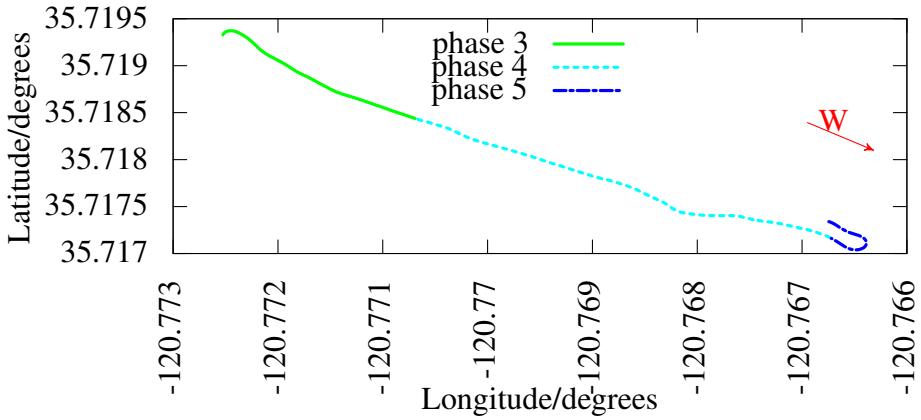


Figure 3.16: Portion of test flight used for hybrid simulation. This is an overhead view of the recorded trajectory used to calculate the observer's true trajectory for the hybrid simulation.

and below the spline fit. This result is due to the fact that the GPS data recorded during a flight is not actually raw data, but instead calculated position data that has been processed using an estimator, such as a Kalman filter, by the GPS receiver module.

The method for using MATLAB functions to compute the cubic spline was inspired by the tutorial in the reference text by Yakimenko [101], and the script used to perform this calculation is shown in Appendix A. In this manner, the smooth piece-wise polynomial was used to calculate the true position of the observer for the simulation, while a look-up table containing recorded times and positions from Snowflake telemetry was used to provide *measured* positions for use by the estimation algorithm. Measurement noise is thus introduced into this simulation due to the difference between the smooth piece-wise polynomial used to calculate the true position of the observer, and the linear interpolation operation of the look-up table used to determine the observer's measured position. This method was used not only for the three position coordinates of the observer, but also for the three angular coordinates (Euler angles) of the observer. For sake of comparison, a separate simulation modeled measurement noise as normally distributed random numbers explicitly

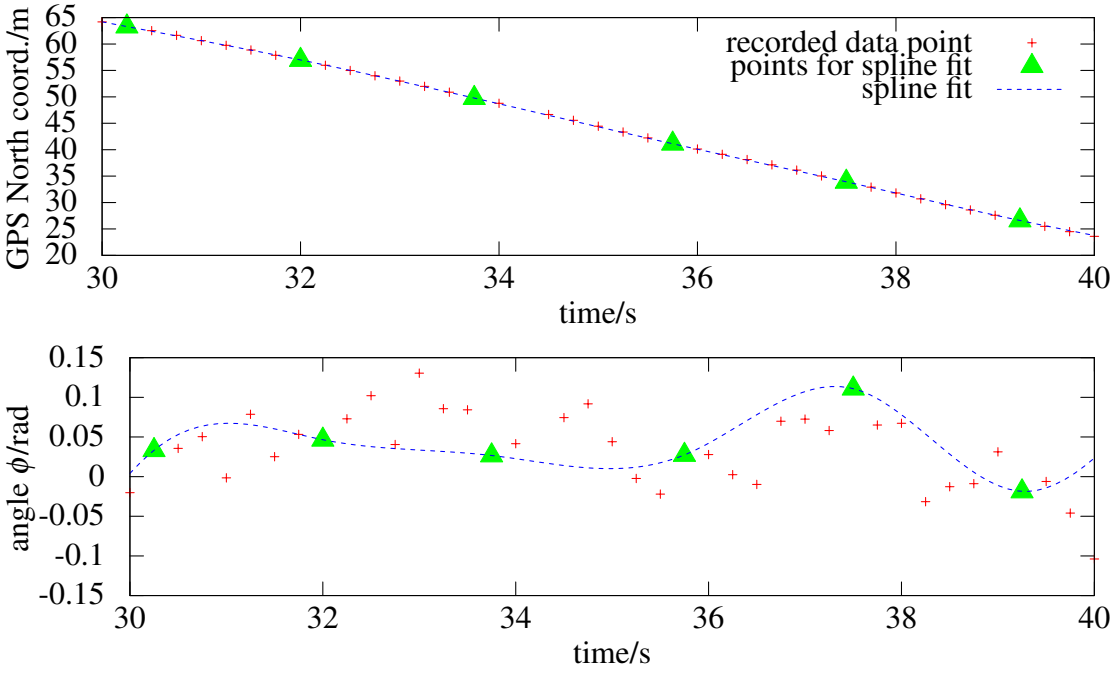
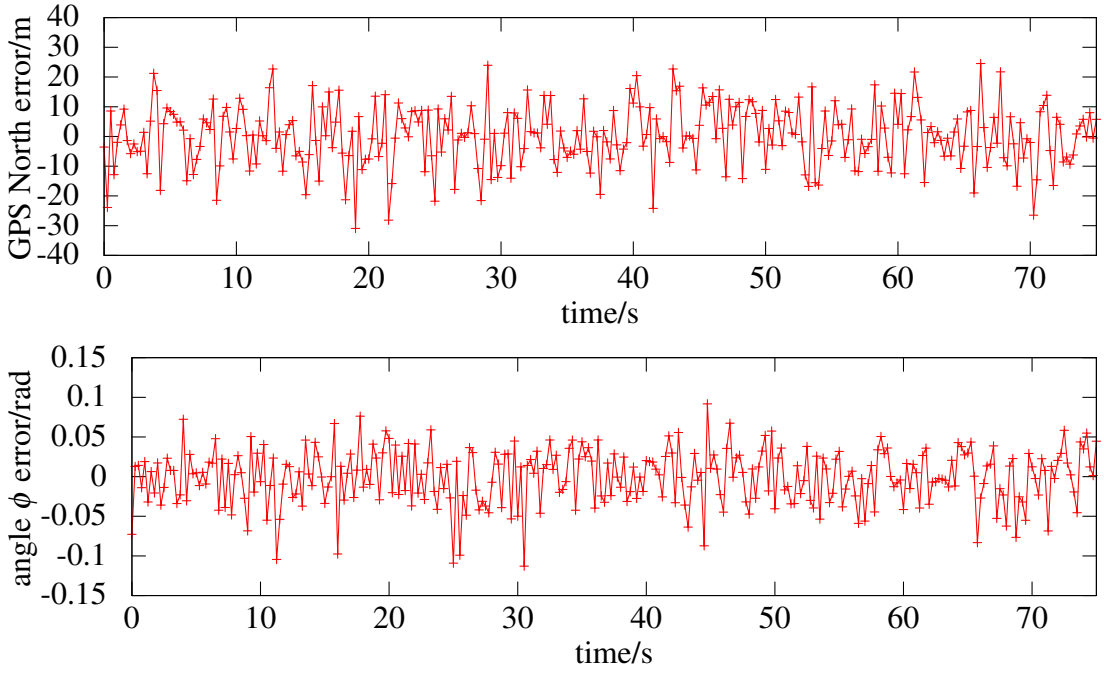
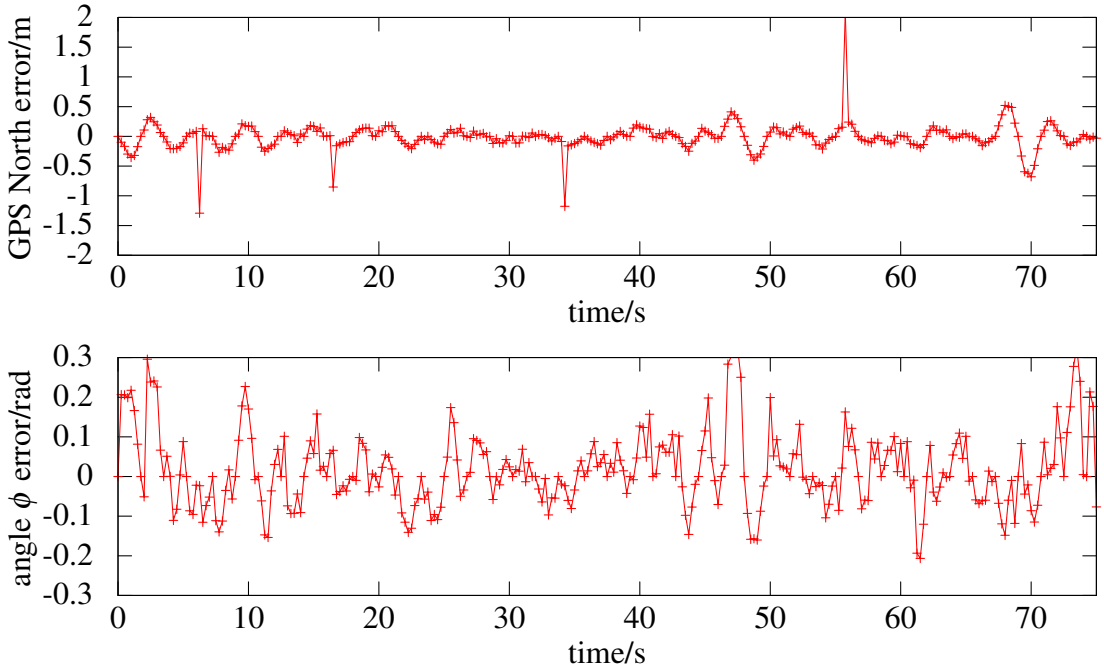


Figure 3.17: Example of spline fitting to flight test data. One out of every seven recorded points was used to calculate the splines. GPS data appears very smooth because it has already been processed by a Kalman filter in the GPS module.

added to the values obtained from the look-up table. Simple additive Gaussian noise is perhaps not realistic for GPS and the accelerometer angle sensors; however, this method allowed fine control over the size of the measurement errors introduced. An example of both the additive Gaussian noise, and the realistic measurement noise obtained from the spline fit are shown in Figure 3.18.



(a) Measurement errors with added Gaussian noise.



(b) Measurement errors using recorded flight data.

Figure 3.18: Two examples of noise modeling. These include: (a) simplistic additive Gaussian noise, and (b) the more sophisticated model-based noise.

3.4.5 Results from Algorithm Verification Model

As described in Section 3.4.1, the first simulation did not use flight test data and instead relied upon completely artificial and simple trajectories for both observer and target. Nevertheless, this simple model was used to realize some findings about the effect of estimator initialization errors and process noise.

Initialization errors for the estimator occur when the assumed initial values of the estimated quantities (position, speed, and course) differ from the true values of those quantities at the beginning of the simulation. Recalling that this first model included *no* measurement noise other than the pixelation of the target image, in the case of position, trial simulation runs with initialization errors from tens of meters to thousands of meters resulted in no appreciable difference in the estimate convergence time. Therefore, the coordinates of the assumed initial position of the target were set arbitrarily to be 5 km different than the true initial coordinates to represent worst-case uncertainty of the target's initial position. For initial speed, when the operational scenario is considered, there is very little reason to assume that the target is moving at a very high speed; therefore, an initial assumption of zero speed represents worst-case uncertainty of the target's initial speed (negative speeds are not considered). For the target's initial course, worst-case initialization error is when the difference between assumed initial course and true initial course is 180°. In this simulation, initial target true course was due South (course 180°), so the assumed initial course was set to be 0°.

The simulation used the initial conditions described above to produce output that is shown as plots of estimation error versus time. Simulation output for the case of simple, straight-line ADS trajectory with no additional sensor noise beside pixelation effects on imaging sensor measurements, is shown in Figure 3.19 and Figure 3.20. These plots indicate that the position and speed estimation errors, respectively, do converge to small values after approximately five seconds during the second period of in-frame observation. The first in-frame observation period of duration approximately seven seconds does not appear long enough to allow convergence. Course estimation error is shown in Figure 3.21, in which it can be seen that the extended Kalman estimation error converges during the first in-frame period, whereas the unscented Kalman estimation error converges at the beginning of the second in-frame period.

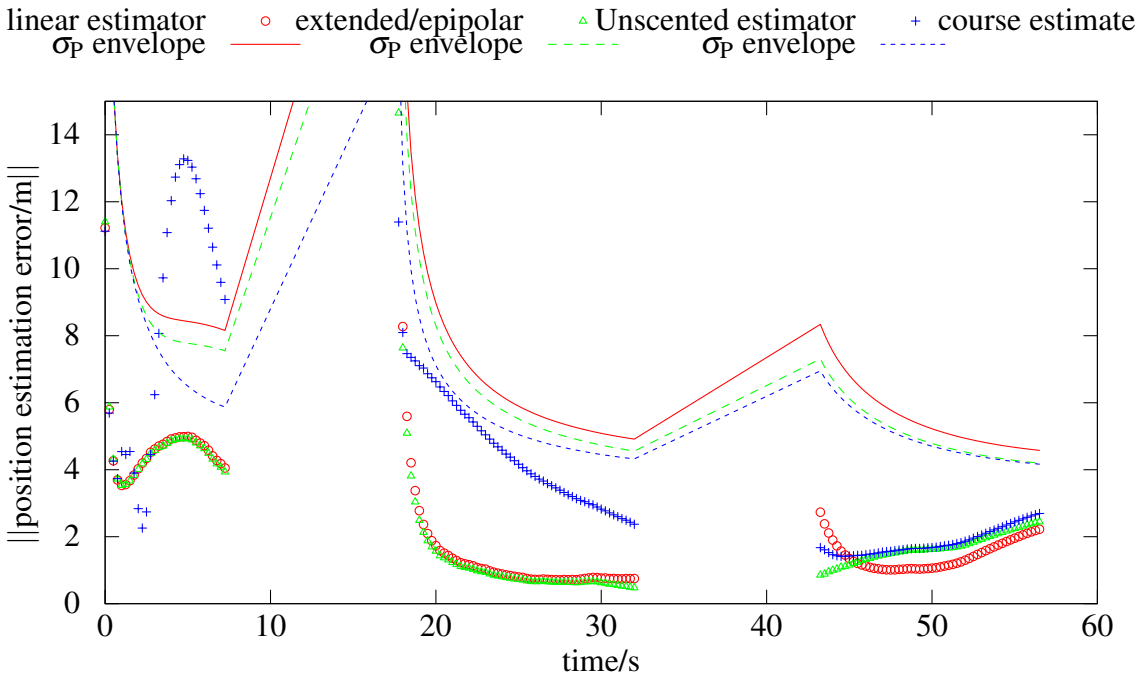


Figure 3.19: Simple trajectory simulation position estimation errors. The norm of the position estimation error vector is shown for three different Kalman estimation methods, along with envelopes representing one σ estimation error.

This first simulation was also used to evaluate the effect of process noise on the Kalman estimation algorithm. In this context, process noise means that the target's motion does not conform to the assumptions made in Sections 3.2.1 and 3.2.4; namely, that the target violates the assumptions of constant speed and course. The system model included process noise consisting of perturbations added to the target's acceleration and turn rate. These perturbations were normally distributed random variables with a zero mean and standard deviations set to representative values: $\sigma_{\dot{V}} = 0.05\,\mathrm{m/s^2}$ and $\sigma_{\dot{\psi}_T} = 0.4\,^\circ\mathrm{/s}$ for acceleration and turn rate perturbation, respectively.

The target's true track, speed, and course is shown in Figure 3.22, along with estimates of those quantities from the extended and unscented Kalman estimators. The longer convergence time of the unscented estimator relative to the extended estimator is clearly visible in this set of plots. Furthermore, it can be seen that even though the target did not maintain constant course and speed, the estimation algorithms were still able to converge to small values of estimation error. Figure 3.22 serves as evidence that the estimation algorithms

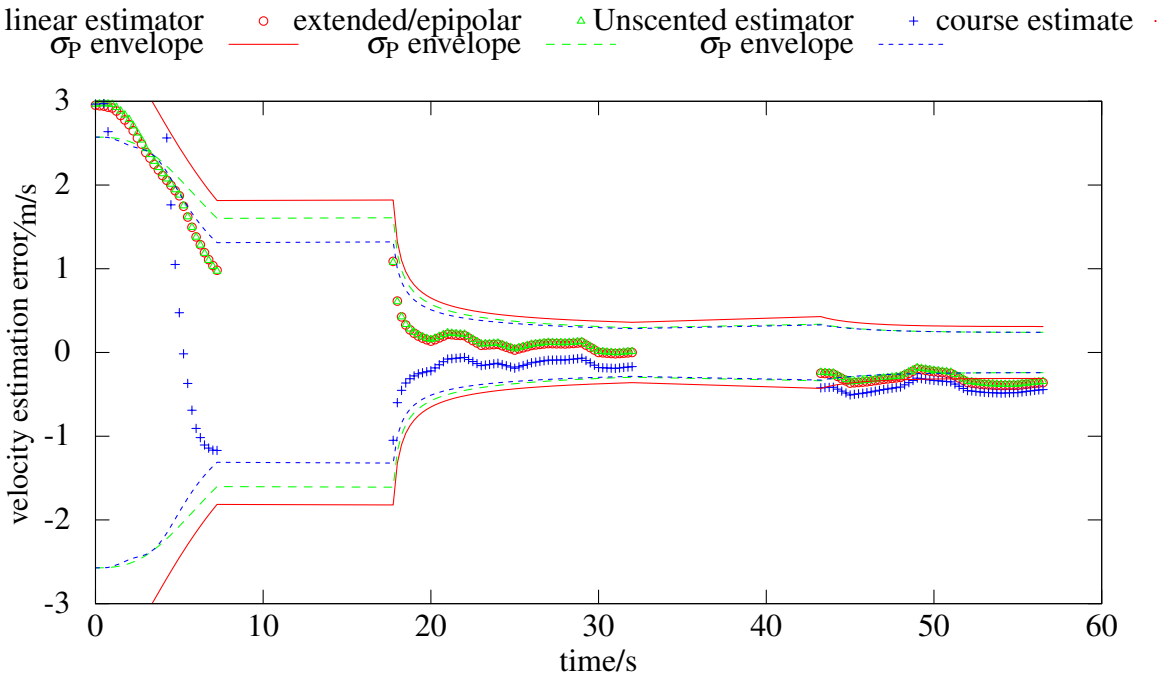


Figure 3.20: Simple trajectory simulation speed estimation errors. Envelopes representing one σ estimation error are shown for three different Kalman estimation methods.

have the capability to reject process noise.

3.4.6 Results from Epipolar Versus DR Comparison

Having gained basic evidence that the model's estimators functioned at a basic level, the next step was determining relative performance among the different estimation techniques. The four separate estimation algorithms described in Section 3.4.2 were analyzed with particular focus on their performance at the instant the target returned to view and immediately afterward. Of the four state variables pertaining to the target, only velocity is not associated with a direct measurement; therefore, special attention was given to how well the four algorithms estimated velocity. To create a challenging test for the estimators, a test that would differentiate their performance, the duration that the target remained out of view was set to a high but realistic value. Test flights of the Snowflake ADS prototype each lasted approximately three minutes, so an out-of-frame duration value of over two minutes was chosen.

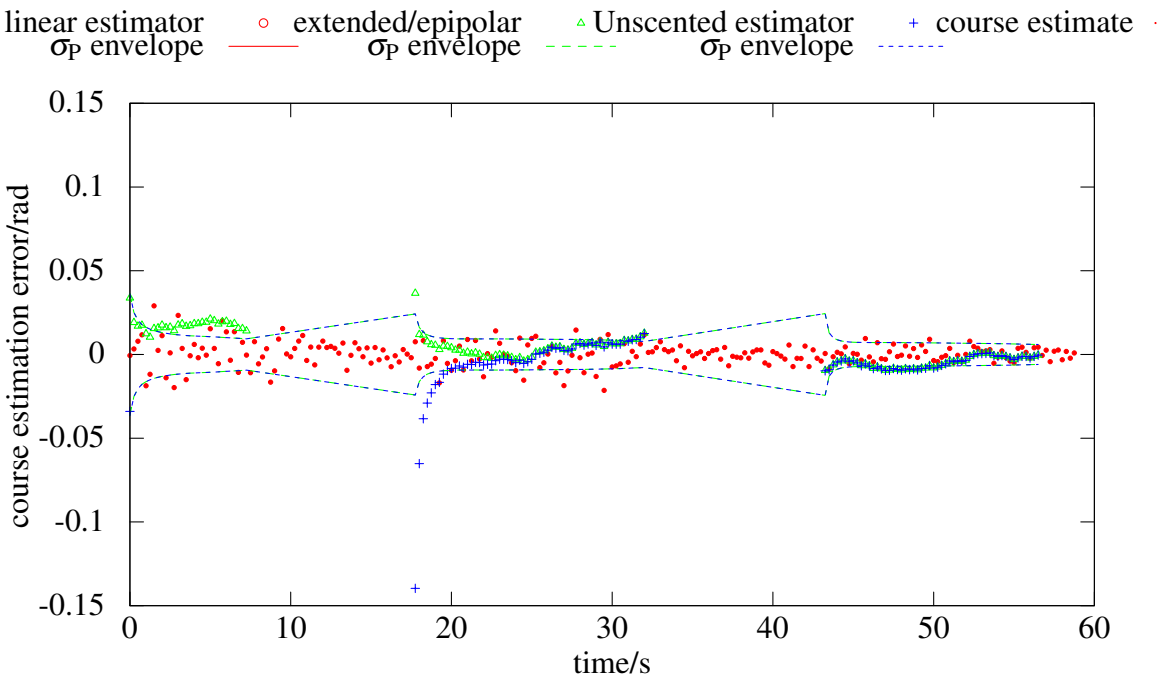


Figure 3.21: Simple trajectory simulation course estimation errors. Envelopes representing one σ estimation error are shown for three different Kalman estimation methods.

The parameter that was varied for this set of simulations was measurement noise. A nominal set of assumed measurement noise standard deviation values was chosen for the physical sensors on the autopilot such as the GPS and the attitude sensors, and the vector of these values was labeled $\bar{\boldsymbol{\sigma}}$. The elements of vector $\bar{\boldsymbol{\sigma}}$ correspond to the assumed standard deviations of the individual noise elements of vector $\mathbf{v}$ from Equation (3.29) in Section 3.2.2. For the simulation runs, both the measurement error covariance matrix $\mathbf{R}$ as well as the actual added measurement noise, were calculated using multiples of the values of the elements of $\bar{\boldsymbol{\sigma}}$.

The first set of simulation runs in this section was conducted while varying the added and assumed measurement noise from zero to $2\bar{\boldsymbol{\sigma}}$ to $4\bar{\boldsymbol{\sigma}}$. For the first run with actual added measurement noise set to zero, measurement error covariance matrix $\mathbf{R}$ was still computed with values of $1\bar{\boldsymbol{\sigma}}$ so that $\mathbf{R}$ would not be a zero matrix. For the first two sets of simulations, actual added process noise was kept at zero to isolate the effect of measurement noise. From the simulation output, position and velocity estimation errors are shown in Tables 3.5 and 3.6, respectively, at the instant that the target returns to view. Along with the

Table 3.5: Position estimation error with long out-of-view duration. The values of standard deviation σ of the estimate uncertainty are from the $\mathbf{P}$ matrix at the instant at which the target returned to view.

Position estimation error and standard deviation/m			
estimator	amount of measurement noise		
	none	$2\bar{\boldsymbol{\sigma}}$	$4\bar{\boldsymbol{\sigma}}$
linear	25.125	377.923	250.234
$\sigma =$	202.712	288.253	348.773
extended/DR	16.264	11.768	32.313
$\sigma =$	27.525	56.608	97.779
extended/epi	14.023	26.938	14.501
$\sigma =$	59.808	92.540	151.764
UKF	36.084	64.555	39.958
$\sigma =$	245.103	315.239	344.008

estimation error value, the standard deviation of the estimate is represented as the square root of the value in the appropriate element of the state estimation error covariance matrix $\mathbf{P}$. Of the estimators compared, one version of the EKF used DR processing as described in Section 3.2.6, while the other three estimators used the epipolar constraint measurement at the back-in-frame instant. The most useful comparisons presented in Tables 3.5 and 3.6 are between the two versions of the EKF and the UKF; the linear estimator is much simpler than the others and therefore was not expected to perform as well.

In Table 3.6, the velocity estimation error of the EKF estimator using the epipolar measurement is lower in all three cases than that of the EKF using DR; however, the computed state estimation error values are higher in two cases than those of the EKF DR estimator. One unexpected feature of the data in Table 3.6 is that the velocity estimation errors of the EKF, epipolar version, did not consistently decrease with increasing measurement noise. This observed behavior could have been due to the specific nature of the random sequences used for measurement noise in the simulation.

To explore further the effect of increasing measurement noise on velocity estimation error, another set of simulations was executed, this time with matrix $\mathbf{R}$ computed with $1\bar{\boldsymbol{\sigma}}$ values, but actual added measurement noise ranging from $5\bar{\boldsymbol{\sigma}}$ to $20\bar{\boldsymbol{\sigma}}$. Velocity estimation errors

Table 3.6: Velocity estimation error with long out-of-view duration. The values of standard deviation σ of the estimate uncertainty are from the $\mathbf{P}$ matrix at the instant at which the target returned to view.

Velocity estimation error and standard deviation/m/s			
estimator	amount of measurement noise		
	none	$2\bar{\boldsymbol{\sigma}}$	$4\bar{\boldsymbol{\sigma}}$
linear	0.128	−2.451	−1.455
$\sigma =$	1.525	2.139	2.578
extended/DR	−0.113	0.125	0.312
$\sigma =$	0.394	0.534	0.805
extended/epi	−0.086	−0.115	−0.016
$\sigma =$	0.304	0.585	1.074
UKF	0.243	−0.454	0.350
$\sigma =$	1.850	2.352	2.558

for these cases are shown in Table 3.7. The velocity estimation errors shown therein for the EKF epipolar estimator are generally increasing with increasing measurement noise, although not monotonically. Considering Tables 3.6 and 3.7 together, it is apparent that the EKF using the epipolar measurement has a smaller velocity estimation error in four of the six cases presented. Clearly, the use of the epipolar measurement conveys some advantage over an estimator that uses only DR.

Judging from the numbers contained in Tables 3.5, 3.6, and 3.7, the UKF seems at first glance to produce inferior estimates to those of the EKF; however, the tables show estimates only at one instant in time, not a history of estimates. A history of estimates from the back-in-view instant to the end of the simulation is shown in Figure 3.23; this figure depicts the velocity estimation error from the same simulation run whose data was used to construct Table 3.6. In Figure 3.23, only the end of the simulation is shown, with the back-in-view instant occurring at 139.5 s of simulation time. The time history of the UKF velocity estimates in this figure suggests that, while the UKF may have higher estimation error at the back-in-frame instant than the EKF estimators, the UKF may converge more quickly than the others. A new set of simulation runs was designed to test this hypothesis.

To isolate the convergence characteristic of the UKF, another simulation was executed, this

Table 3.7: Velocity estimation error with high measurement noise. The values of standard deviation σ of the estimate uncertainty are from the $\mathbf{P}$ matrix at the instant at which the target returned to view.

Velocity estimation error and standard deviation/m/s			
estimator	amount of measurement noise		
	$5\bar{\boldsymbol{\sigma}}$	$10\bar{\boldsymbol{\sigma}}$	$20\bar{\boldsymbol{\sigma}}$
linear	−13.769	−28.814	−62.764
$\sigma =$	1.507	1.494	1.496
extended/DR	0.185	0.393	0.942
$\sigma =$	0.438	0.434	0.419
extended/epi	−0.322	−0.631	−0.609
$\sigma =$	0.306	0.333	0.372
UKF	−6.376	−11.643	−16.456
$\sigma =$	1.848	1.841	1.785

time with the actual added measurement noise set to zero. The time history of velocity estimation errors for this case is shown in Figure 3.24. In this figure, it is clear that the first back-in-view UKF velocity estimate error at simulation time 139.5 s is larger than those of the other estimators; however, the UKF estimation error converges quickly toward zero. This convergence occurs more quickly than the convergence of the estimation errors from the EKF estimators. A repeat of this simulation run was conducted, this time with process noise added with the same values as described in Section 3.4.5, and the resulting velocity estimation errors are shown in Figure 3.25.

The superior convergence characteristic of the UKF over the EKF for these simulations is evident when considering Figures 3.23, 3.24, and 3.25 together. Clearly, the use of the UKF, along with the use of the epipolar constraint, confers an advantage to an estimator in the cases represented by the aforementioned simulations.

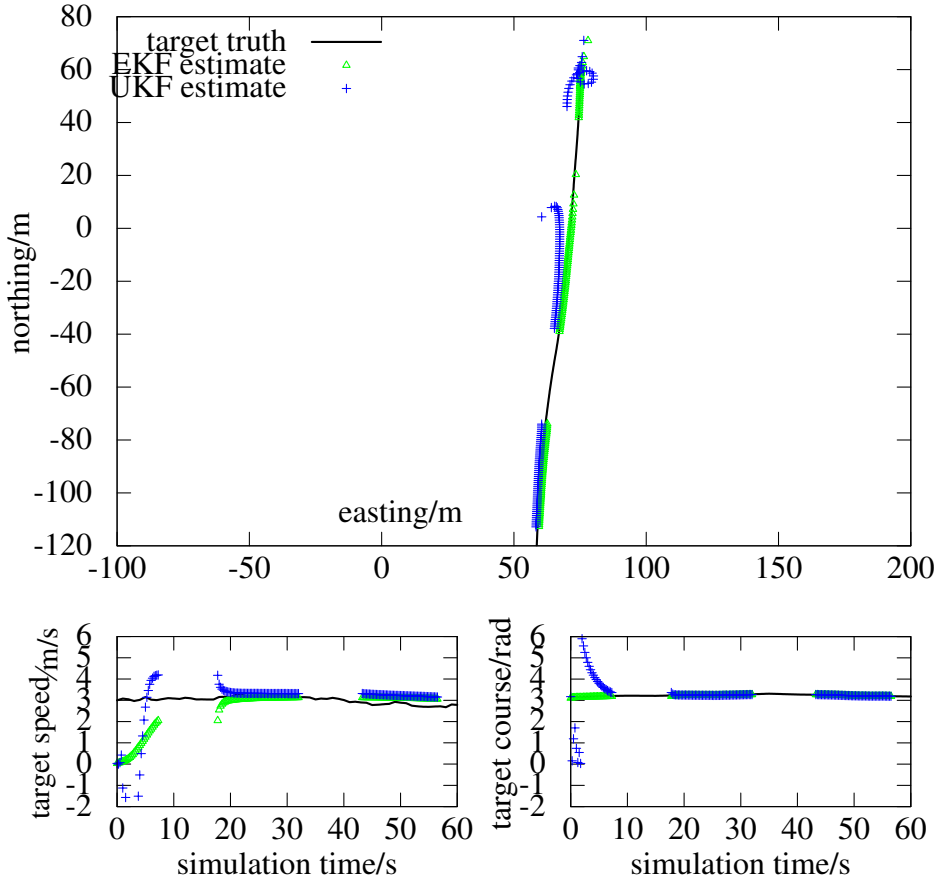


Figure 3.22: Simple trajectory simulation estimated target track. These estimates were computed using two different Kalman estimation methods: the EKF and UKF.

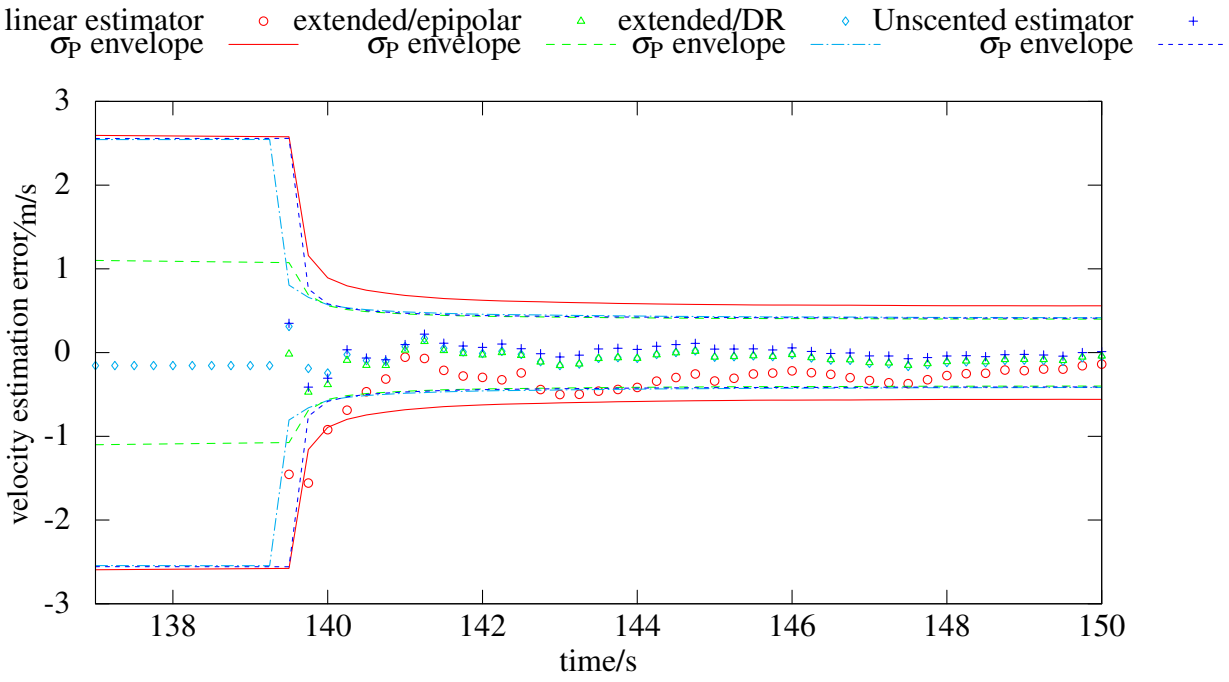


Figure 3.23: Velocity estimation error after target returns to view. For this simulation, the target was out of view until 139.5 s of simulation time.

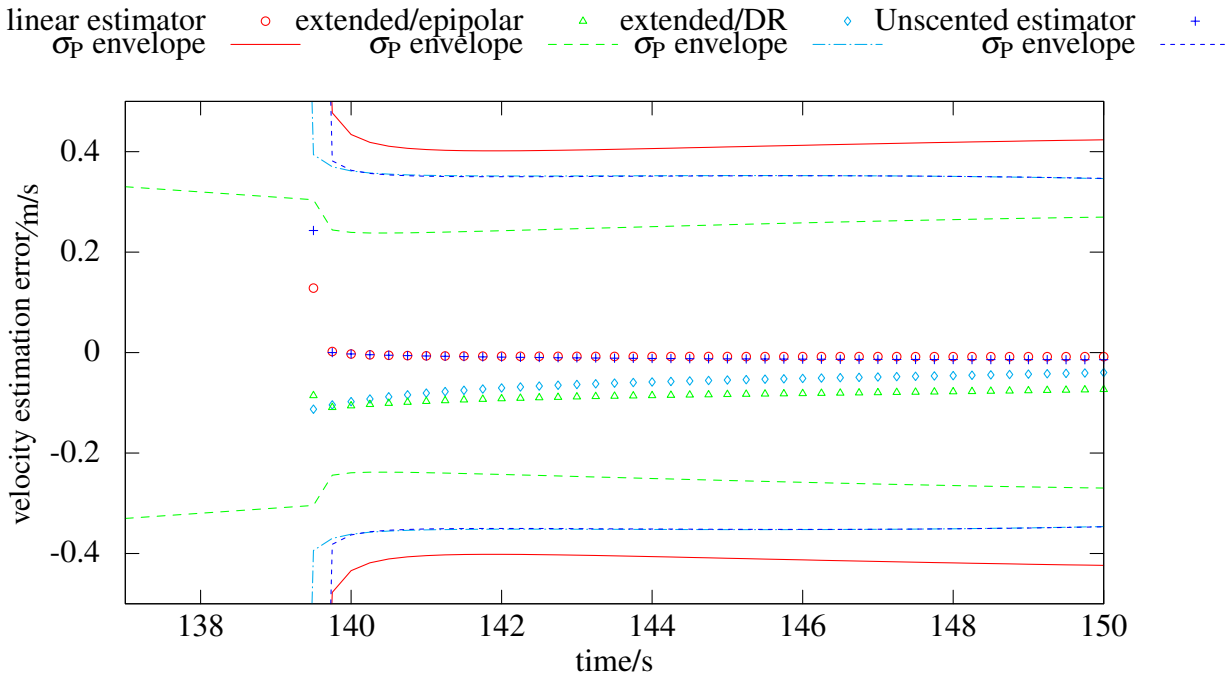


Figure 3.24: Velocity estimation error with no noise. Both process noise and measurement noise are set to zero in this simulation run.

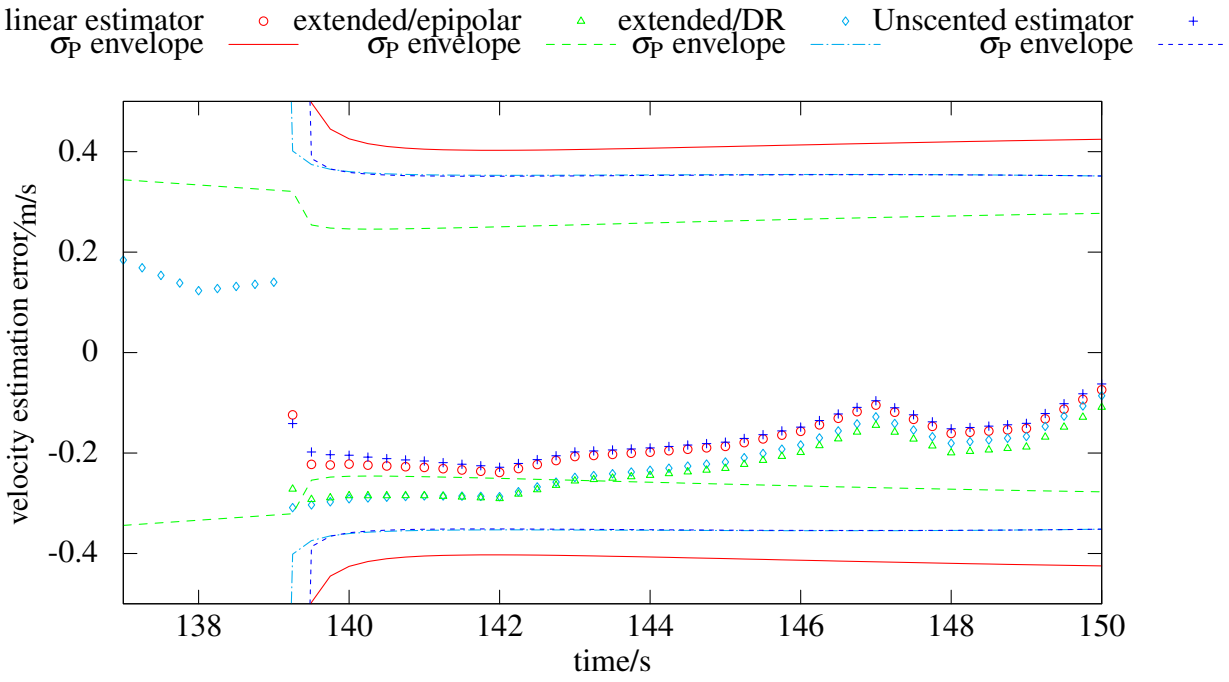


Figure 3.25: Velocity estimation error with process noise. Process noise is set to $\sigma_{\dot{V}} = 0.05\,\mathrm{m/s^2}$ and $\sigma_{\dot{\psi}_T} = 0.4\,^\circ/\mathrm{s}$ in this simulation run.

3.4.7 Results from the Trade Study

All further simulations were conducted using the hybrid model described in Section 3.4.4. These simulations comprise the trade study. The first runs of the revised hybrid simulation were made to validate that the estimation algorithm was functioning properly. To validate correct operation, simulation parameters were set so that there was no external measurement noise injected into the algorithm. Aside from pixelation noise described in Section 3.4.1, the observer had perfect knowledge of its own position and orientation throughout the simulation. The estimator parameters for this run were from the *nominal* parameter set from Table 3.4.

The estimation error for target position, speed, and course for the case without external measurement noise are shown in Figure 3.26. Examining Figure 3.26a, it was concluded that the estimator was functioning correctly, and that convergence to 1 m estimation error in position occurred within 25 s. Convergence to within 1 m/s estimation error in speed occurred within 15 s, as shown in Figure 3.26b. In this plot, the initial speed estimation error was approximately 1.5 m/s, a fairly small value, due to the worst-case initial assumption of zero speed as explained at the beginning of this section. The estimation error in target course converged to less than 0.05 rad in 8 s, as shown in Figure 3.26c.

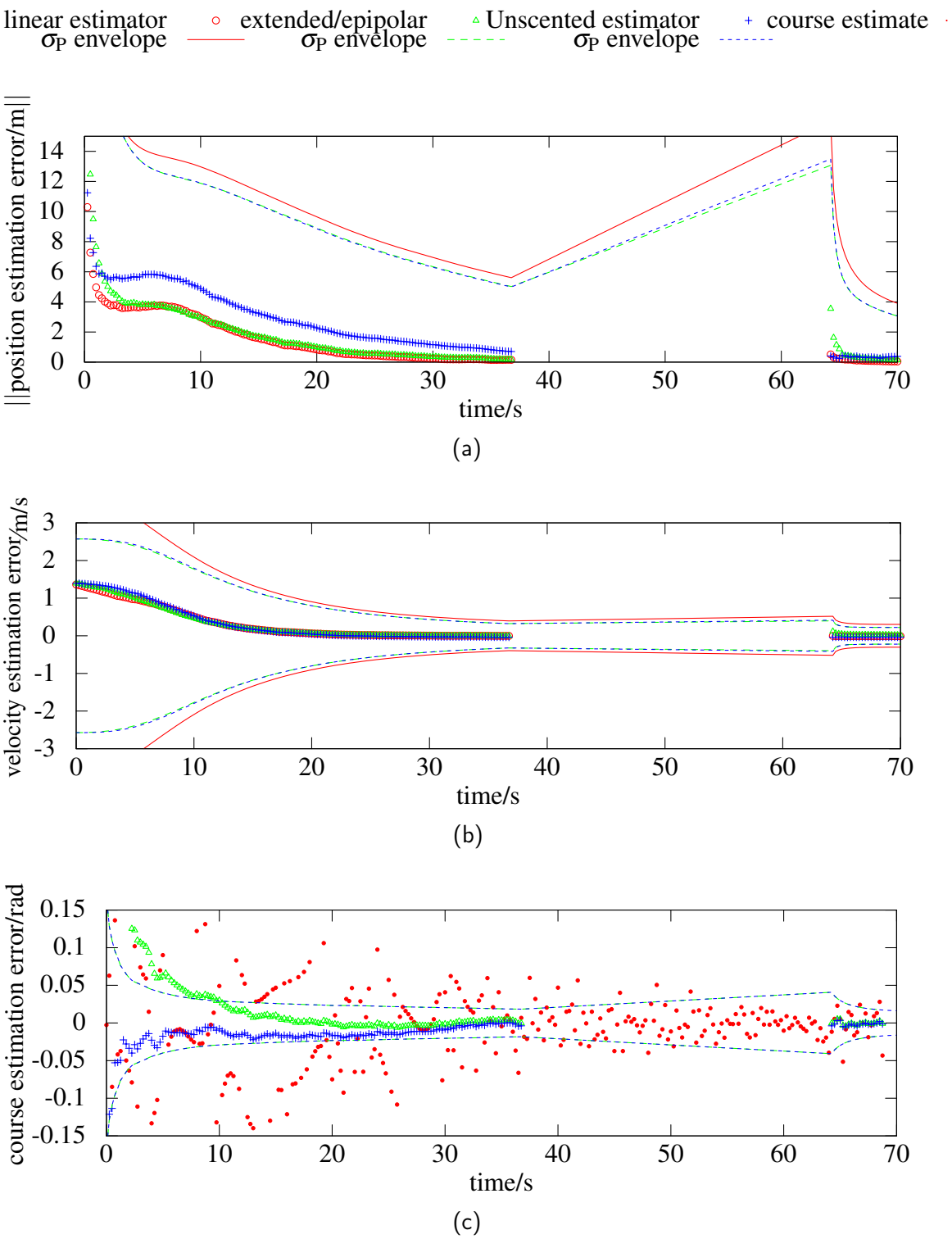


Figure 3.26: Model validation estimation error plots. These include: (a) position, (b), velocity, and (c) target course versus simulation time.

The next simulations included additive white noise to represent measurement noise for observer position and orientation data. The estimator parameters were those from the improved values listed in Table 3.4. The goal of this simulation run was to demonstrate the effect of the incorporation of measurement noise given the best available estimator performance.

The estimation error for target position, speed, and course are shown in Figure 3.27. From these plots, it was concluded that the estimation errors still converge to small values; however, the effect of the measurement noise can be clearly seen, especially after 35 s of simulation time in the position error plot, Figure 3.27a, and after 20 s of simulation time in the speed error plot, Figure 3.27b. The improvement from the estimation algorithm over the raw measurements in the course estimation plot shown in Figure 3.27c is striking.

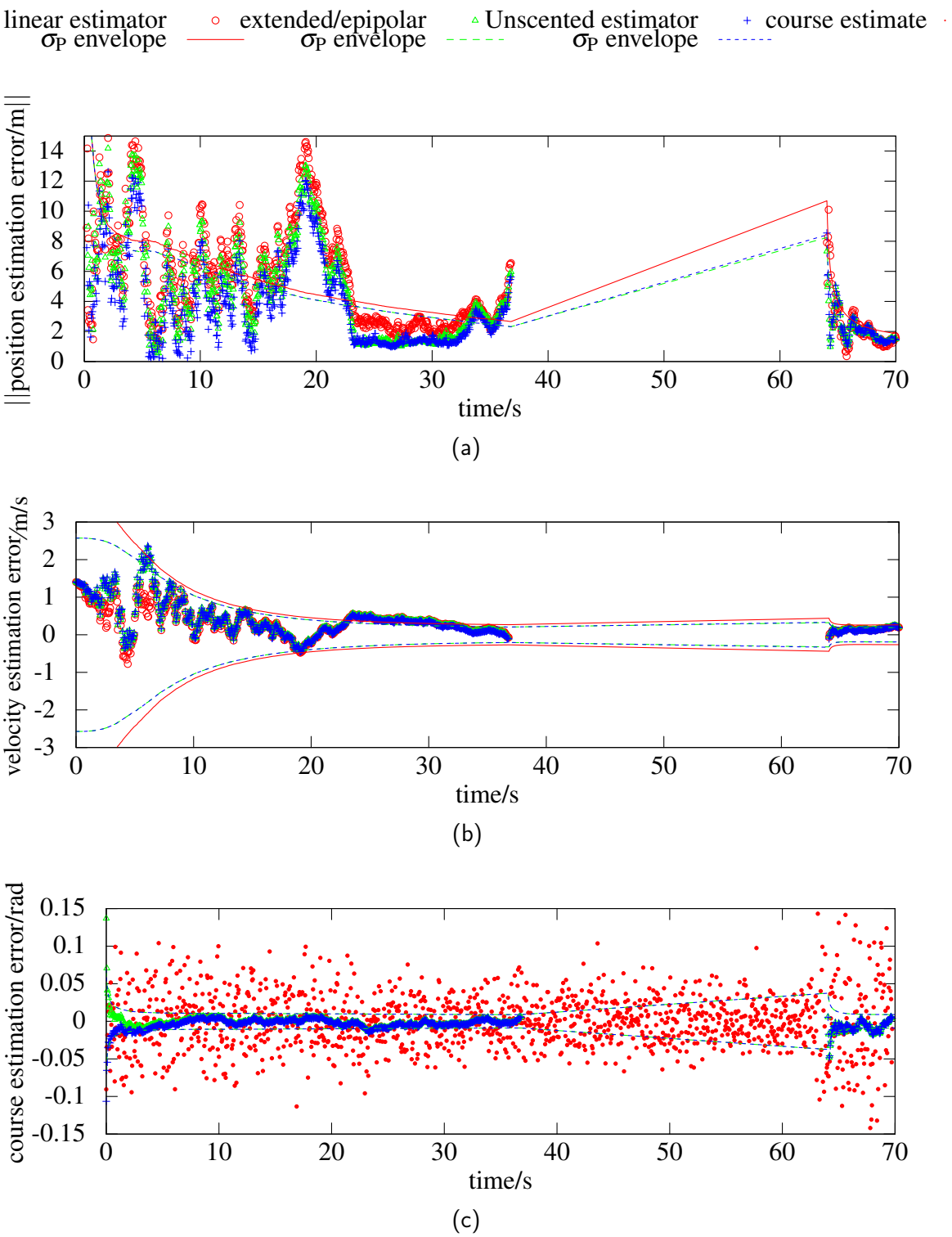


Figure 3.27: Best parameter set simulation estimation errors. These include estimation errors in (a) target position, (b) speed, and (c) target course, all plotted versus simulation time.

The next step was to return the estimator parameters to the nominal values while maintaining the same type and level of measurement noise.

In comparing the plots of Figure 3.28 with the previous set in Figure 3.27, it is apparent that the most significant differences in performance are seen in estimation of target position. Comparing the position estimation error plots in Figure 3.27a and in Figure 3.28a, the estimation error between 20 s to 30 s is below 5 m for the improved estimator parameters, and the estimation error is almost 10 m during the same span of time using the nominal estimator parameters. It is also very interesting to notice that in both sets of plots, those of Figure 3.27 and of Figure 3.28, the Kalman estimator using the Unscented Transformation, or the UKF, had very similar performance to the extended Kalman estimators that used derivative-based techniques to calculate measurement error covariance.

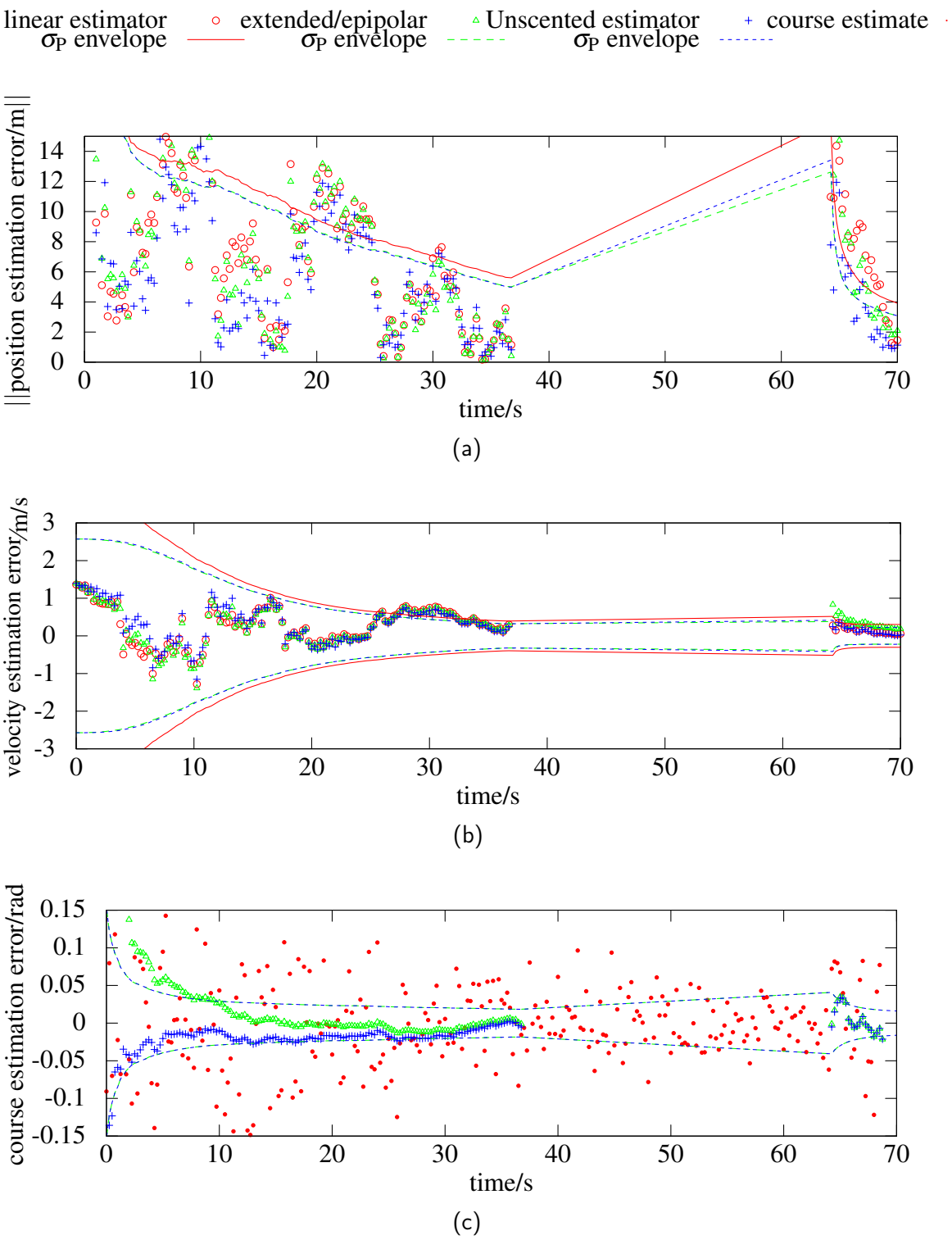


Figure 3.28: Nominal parameter set simulation estimation errors. These include estimation errors in (a) target position, (b) speed, and (c) target course, all plotted versus simulation time.

Given the differences in performance between the previous two simulation runs with identical measurement noise, the next step was to determine which of the estimator parameters, estimator sampling rate, or image plane size, had the greater effect on estimator performance. For the next simulation run, the image plane size was not changed, but the estimator sampling rate was increased to 20 Hz.

In comparing the set with the 20 Hz sampling rate only, known as the *fast* parameter set of Figure 3.29, to that which had both the higher sampling rate and also better image plane resolution, known as the *improved* parameter set of Figure 3.27, there is not much difference between the two sets of plots except for the estimation of target course, shown in Figure 3.29c, compared to Figure 3.27c. The estimation error for target course converged to small values about 10 s faster for the estimator with the improved parameter set that had both the faster sampling time and finer image plane resolution and whose performance is shown in Figure 3.27c. A logical inference is that the majority of the improvement of estimator performance over the case with nominal parameters (Figure 3.28) came from increasing the sampling time, and that the increase in image plane resolution mainly led to improved performance in target course estimation. Therefore, it was concluded that the first, most effective improvement that should be made to the estimation algorithm is an increase to its sampling rate.

The last step in the trade study was to examine the effect of a more realistic model of measurement noise. For this simulation, the additive Gaussian noise was replaced by a noise model that generated measurement noise using the piece-wise polynomial (cubic spline) method described in Section 3.4.4. This simulation was based on the fast parameter set of Table 3.4 because the fast set represents the next likely improvement in Snowflake avionics.

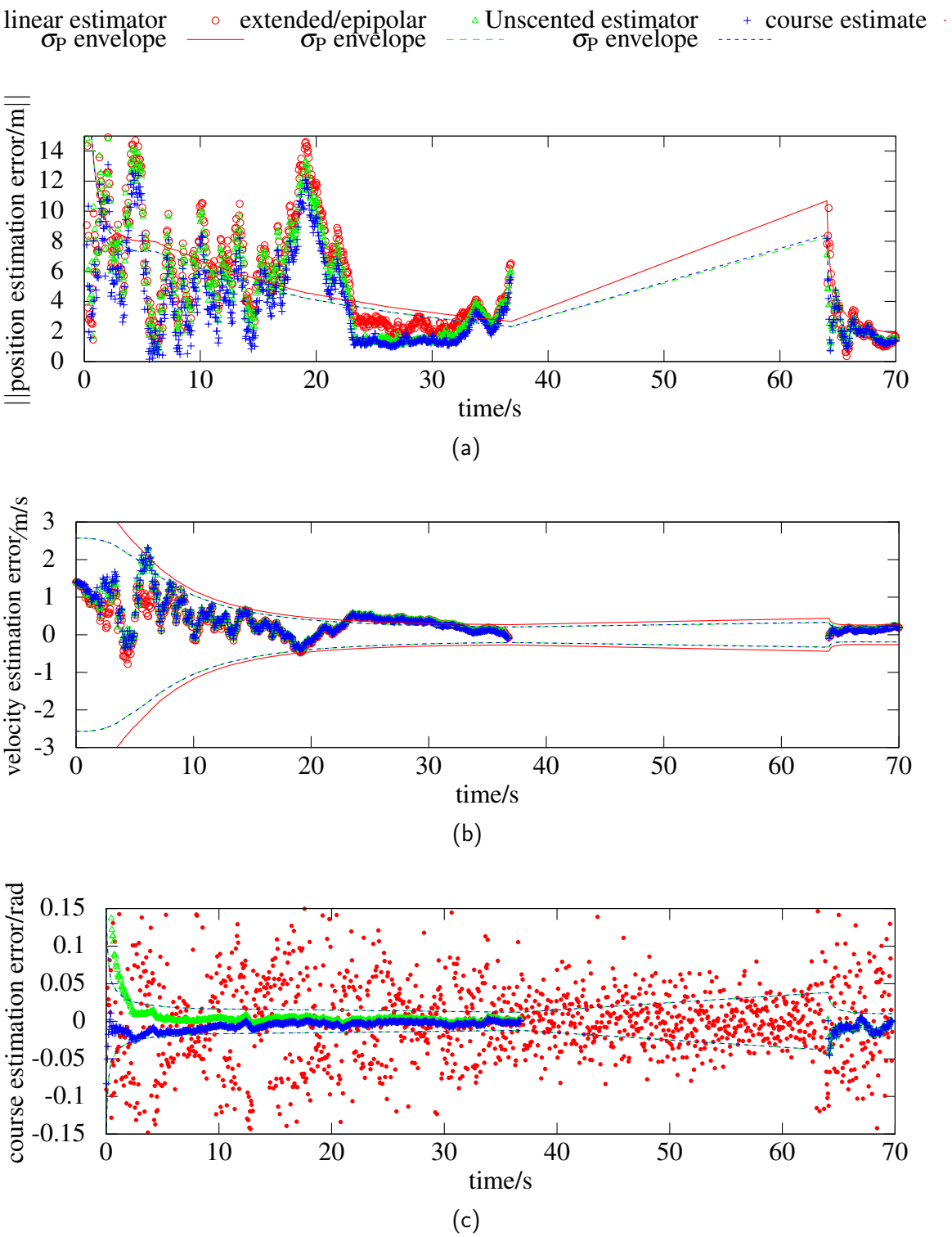


Figure 3.29: Fast parameter set simulation estimation errors. These include estimation errors in (a) target position, (b) speed, and (c) target course, all plotted versus simulation time.

When analyzing the plots of Figure 3.30, the fact to keep in mind about the revised noise model is that it does contain time-varying biases, and therefore violates one of the prime underlying assumptions of Kalman estimation. This assumption is that process noise and measurement noise each have zero mean. Increased estimation error is indeed observed in this set of plots with this realistic noise model in Figure 3.30, relative to the set of plots with the Gaussian noise model and the same fast parameter set in Figure 3.29.

Errors in the estimate of target position shown in Figure 3.30a decrease from their initial errors to a minimum at around 18 s of simulation time, but then start to increase before the target leaves the field of view at a simulation time of 37 s. Errors in the estimate of target speed shown in Figure 3.30b show some initial oscillation, then reach a minimum at 18 s to 19 s of estimation time, then increase. Of all the plots in Figure 3.30, the estimation of target course in Figure 3.30c displays a consistent convergence to small values of error and represents an improvement over the raw measurements alone.

Two issues arise when analyzing this last set of simulation results. First, the hybrid simulation run in this case does not have a long enough time duration to give a more accurate picture of the performance of the estimation scheme in the presence of non-Gaussian measurement noise. Second, the investigation of what is realistic measurement noise for GPS and MEMS accelerometer and turn rate sensors is an entire topic itself. The simulation performed here is just an initial look at how the estimator might perform if the measurement noise is not Gaussian, and the results that the simulation provides do not constitute a conclusive answer to the question of performance in the presence of non-Gaussian noise. Overall, the estimation errors plotted in Figure 3.30, suggest that the estimator generally displays convergent rather than divergent error behavior; therefore, it is concluded that this estimation method should be further studied in flight test.

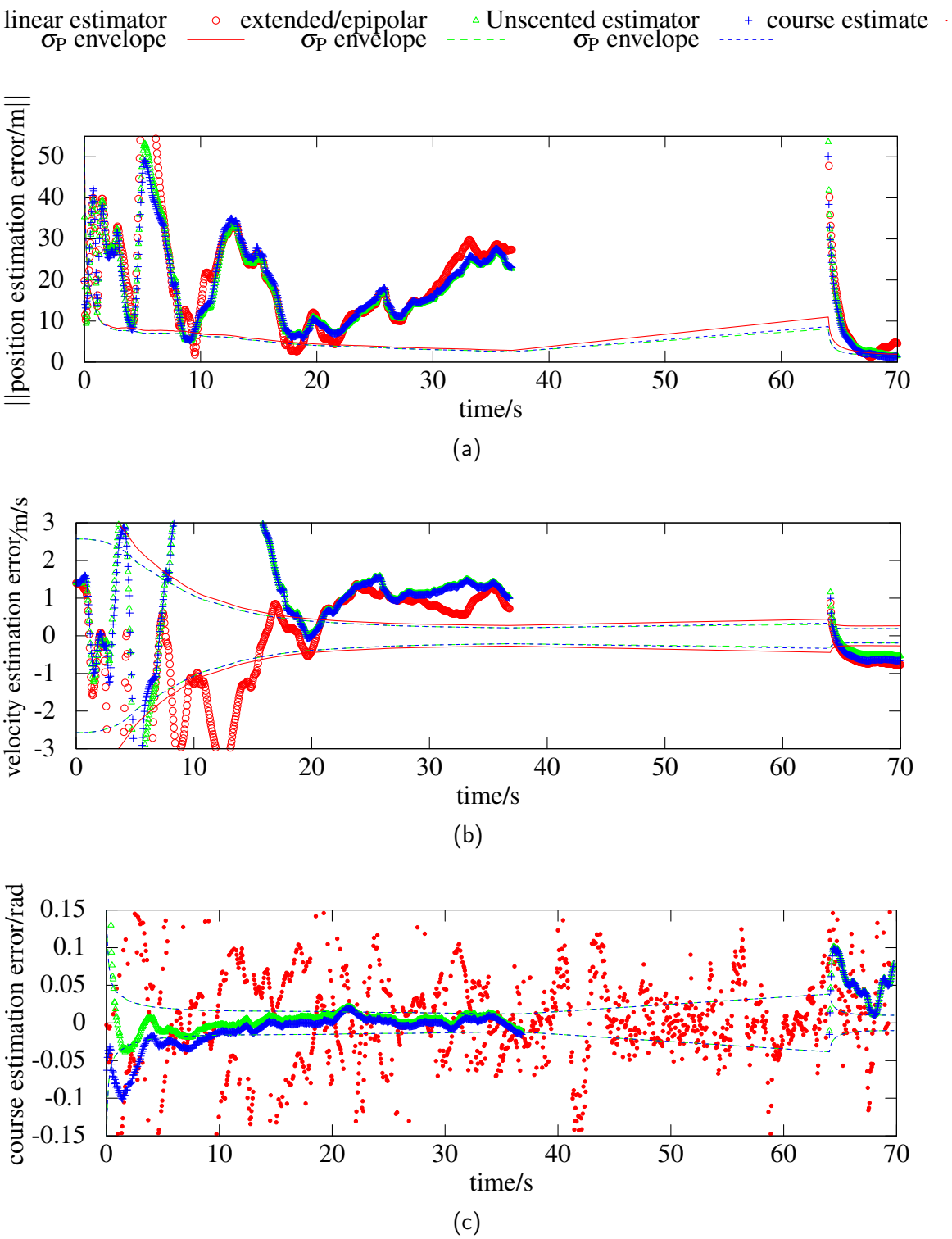


Figure 3.30: Simulation estimation errors with realistic noise. These include estimation errors in (a) target position, (b) speed, and (c) target course, all plotted versus simulation time.

CHAPTER 4:
Wind Estimation and Wind Profile Modeling

Because this research is focused on enabling an ADS to land on a moving platform, landing accuracy is even more vital as a performance goal. In the previous chapters, methods have been considered for an ADS to estimate the position and velocity of the moving landing platform on the surface. As important as estimating the target's state is, perhaps even more important is estimating the wind environment through which the ADS must fly to the target. In this chapter, a new method will be presented for developing a model of the wind profile that, while it is more complex than those models used currently in the field of ADS GN&C, it more closely matches the observed characteristics of wind profiles in nature. Through simulation and flight test, it will be demonstrated that this improved wind modeling method enhances the accuracy of an ADS attempting to land on a moving platform.

4.1 Understanding the Wind Environment

Chapter 1 introduced the importance of wind knowledge to the task of performing an aerial delivery. Now, it is time to analyze in detail various ways that an ADS in flight can learn about the wind environment through which it is flying. First, though, a review of terminology is helpful. Section 1.1.3 introduced the concept of a functional relationship between the average horizontal wind velocity and height about the earth's surface. This relationship is called the *wind profile model*. Section 2.5.2 mentioned methods of measuring the wind velocity with dedicated sensors and also discussed the current simple methods that ADSs use to incorporate wind profiles into their guidance. In that short discussion in Chapter 2, an important distinction was introduced between the computation of wind at the ADSs current altitude, and the prediction of wind magnitudes at altitudes below the ADS. The term wind profile modeling was introduced and defined in Section 1.1.3, and that term will be used in the same way. Introducing the term *wind estimation*, the terminology that will be used in this chapter is:

wind estimation using on-board sensors, and possibly also performing specific maneuvers, the ADS constructs a value for the horizontal, and possibly vertical, wind velocity at the current altitude

wind profile modeling constructing a mathematical functional relationship $W(z)$ for wind velocity in the air mass at all altitudes between the surface target and the airborne vehicle.

4.1.1 Methods of Wind Estimation

With the terms wind estimation and wind profile modeling now defined, a quick review of Sections 1.1.3 and 2.5.2 will confirm that these two previous sections covered wind profile modeling but not wind estimation. Wind estimation has been a topic of study not only in the field of autonomous aerial delivery systems (ADSs), but also in the field of small, fixed-wing UASs. Typical formulations of the wind estimation problem for ADSs are given by Carter [38] and Bergeron [48]. For fixed-wing UASs, Petrich and Subbarao [102] describe a straightforward Kalman estimation approach.

To formulate the wind estimation problem here, the development in a recent article by Ward, Costello, and Slegers [103] will be used as a guide. Consider the wind triangle depicted in Figure 4.1. The origin of the coordinate system in Figure 4.1 is located at the center of gravity of the ADS which is viewed from directly above. Assuming that the ADS flies with constant velocity relative to the air-mass (constant airspeed) and that this airspeed velocity vector is aligned perfectly with the longitudinal axis of the ADS (no sideslip), then one side of the triangle, the *air vector*, is defined by the ADS's true heading ψ and its true airspeed. The *wind vector*, here representing the horizontal velocity of the air-mass with respect to an earth-fixed reference frame, is added to the air vector to yield the *ground vector*. The ground vector is the velocity of the ADS with respect to the earth-fixed reference plane. It is defined by the ADS's track and ground-speed and is commonly measured by an on-board GPS receiver.

The wind estimation problem is this: given a measurement of the ground vector, and sometimes partial knowledge of the air vector as well, such as knowledge of the true heading angle only, compute an estimate of the wind vector. Without full knowledge of both the ground vector and the air vector, this is an under-determined problem. To solve the problem, the ADS must gain information so that the air vector is known. Carter achieves this by assuming prior knowledge of the glide ratio of the ADS, then computing airspeed from the measured vertical velocity [38]. This algorithm assumes that the ADS's true heading is

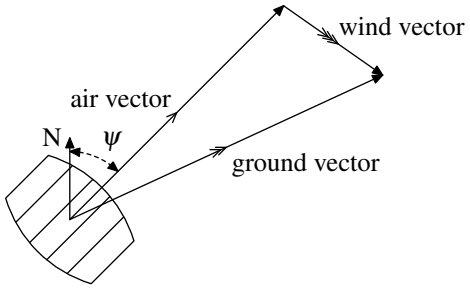


Figure 4.1: The wind triangle. The air vector is shown originating from an overhead view of an ADS.

available for measurement. The algorithm of Ward, Costello, and Slegers, by contrast, relies on having multiple measurements of the ground vector along multiple different tracks, then solving a linear regression problem to compute the two orthogonal components of the wind vector [103]. Using this method, the true heading of the airspeed vector is not needed; however, it is necessary that the tracks used in the computation be enough in number and wide enough in span to ensure that the linear regression is well-conditioned.

These two wind estimation methods each estimate the wind vector in a two-dimensional plane. A much simpler version of the method by Ward, Costello, and Slegers allows computation of the magnitude of the wind vector in one dimension. In the one-dimensional algorithm, the ADS flies two tracks in opposite directions and computes the mean of the measured ground-speeds. This mean value is the airspeed which can be used to compute the wind speed along the axis of the flown segments. This method is the one used by the Snowflake prototype ADS and is discussed in detail in Section 4.3.

4.1.2 External Sources of Wind Information

Of course, the wind estimates obtained by the methods outlined in Section 4.1.1 are valid only for the current altitude of the ADS. More precisely, when an ADS is using a method that requires multiple measurements, the ADS is descending while taking the successive measurements so that the resultant computed wind vector should be associated with the range of altitudes traversed.

For effective trajectory planning, the ADS needs a wind profile, which is a family of wind estimates over a range of altitudes. The ADS could receive this wind profile information from a dropsonde or a balloon as described in Section 2.5.2; these devices are merely per-

forming wind estimation at successive altitudes as they either float down or float up. An ADS that had completed its flight could also organize its recorded wind estimates into a wind profile and transmit that to a following ADS still in flight. As an alternative, recent work by Hermann proposed using a dedicated sensor such as a light detection and ranging (LIDAR) to take direct measurements of the horizontal wind velocity at various altitudes and transmit them to a descending ADS. One recent initiative from Air Force Research Laboratory (AFRL) combines these two ideas by proposing to drop a palletized LIDAR wind profile measurement system from an aircraft over the delivery area. During its descent, the LIDAR would then measure and transmit the wind profile back to the aircraft [104].

4.1.3 Methods of Wind Profile Modeling

The operational situation in which the ADS will be employed may not be conducive to using an external source of wind profile information; therefore, having an on-board means of determining this profile is prudent. This on-board wind profile is really just a guess about how the magnitude and direction of the wind vector would change over a range of altitudes from the ADS to the surface. As mentioned in Section 1.1.3, these models take the form of a functional relationship of horizontal wind speed versus height above the surface of the earth. A simplistic assumption, but one that is easy to use in computations, is that the wind does not change with height: this is the constant wind profile assumption. A variation of this profile is one in which there are several sections, each one a constant profile, but each with a different horizontal wind magnitude. This is the piece-wise constant wind profile. Examples of a constant and a piece-wise constant wind profile are shown in Figure 4.2. Note that for these plots of wind profile, the independent variable, height above the surface, is plotted on the vertical axis.

There are many other possible wind profile models, including linear, logarithmic, and power law profiles; two of these examples are illustrated in Figure 1.5 in Section 1.1.3. Next, we should examine the basis upon which a particular wind profile should be chosen over others.

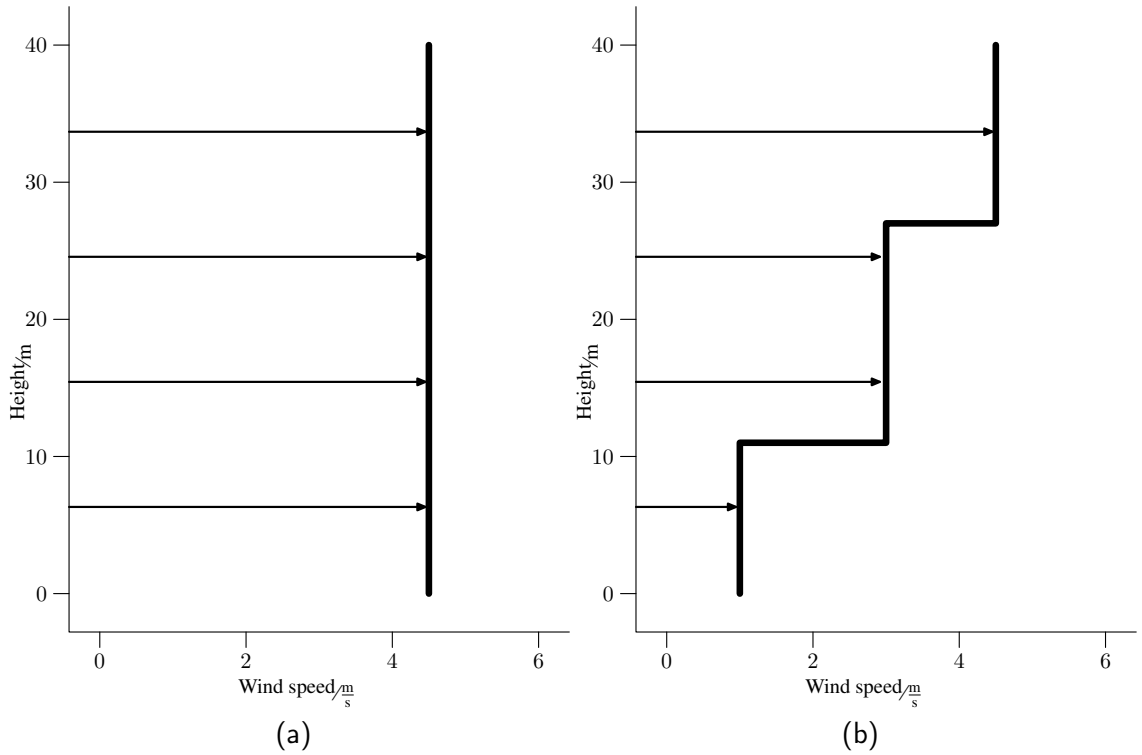


Figure 4.2: Constant (a) and piece-wise-constant (b) wind profiles. Wind profiles are traditionally plotted with the independent variable, height above the surface, plotted on the vertical axis.

4.1.4 Implications of Atmospheric Boundary Layer Theory

The field of meteorology contains a theory that offers insight into the shapes of horizontal wind profiles under different conditions. A basic text by Salby introduces the atmospheric boundary layer (ABL) theory, also called the planetary boundary layer theory, which contains methods for predicting smooth and turbulent airflow in the layer of atmosphere closest to the earth's surface [12]. According to this theory, the ABL is defined as the lower-most portion of the troposphere extending from the surface to as high as 4 km in height. Within the ABL, there exists a lower layer within which heat and moisture interactions between the atmosphere and the Earth's surface cause significant changes in wind speed with height. Another basic text by Stull names this lower layer the *surface layer* and describes it as extending from the surface of the earth up to approximately 5 % of the boundary layer height; therefore the surface layer height is typically in the range 20 m to 200 m [13].

When an ADS makes an assumption that the wind profile from its current altitude to the surface is constant, this assumption yields a mathematical problem for trajectory planning that is much more tractable; however, this assumption has no basis in meteorological boundary layer theory. In fact, constant wind profile assumptions used by ADSs violate one of the fundamental assumptions of ABL theory: the no-slip condition. This condition stipulates that the horizontal wind velocity at the earth's surface is zero [12]. In further contrast to the constant wind profile assumption, fundamental atmospheric boundary layer theory predicts a logarithmic increase of wind speed with height in the surface layer as in the following relation:

$$W_2 = W_1 \frac{\ln \frac{z_2}{z_0}}{\ln \frac{z_1}{z_0}} \tag{4.1}$$

where W_1 and W_2 are wind speeds at heights z_1 and z_2, and parameter z_0 is known as an *aerodynamic roughness length*. The value for the aerodynamic roughness length at the surface is estimated empirically based on the ground terrain. In particular, this parameter has a very small value over the sea surface, and is fairly constant, and therefore easy to estimate [13].

The work by Stull [13] presents several cases of overall atmospheric conditions, with corresponding mathematical models for the general wind profiles under these conditions. The simplest wind profile, the logarithmic model, corresponds to the case of neutral atmospheric stability. Neutral stability refers to the tendency of a parcel of air, once displaced vertically from its original position, to remain in its new position, neither continuing vertical motion, nor returning to its original position. It will be assumed in this chapter that the logarithmic model corresponding to neutral surface layer atmospheric stability, prevalent on overcast and windy days, is the appropriate wind profile model to use in conjunction with aerial delivery. Thus, according to this theory, the functional dependence of wind speed on height can be plotted as a straight line only on a plot with a logarithmic scale on the axis on which height is plotted. Figure 4.3 contains a plot of the logarithmic wind profile that illustrates this relationship graphically.

In this chapter, it will be shown that the use of a logarithmic model of near-surface boundary layer winds enables a trajectory planning algorithm to create trajectories that result in more accurate landings. The price paid for improved accuracy is increased computational

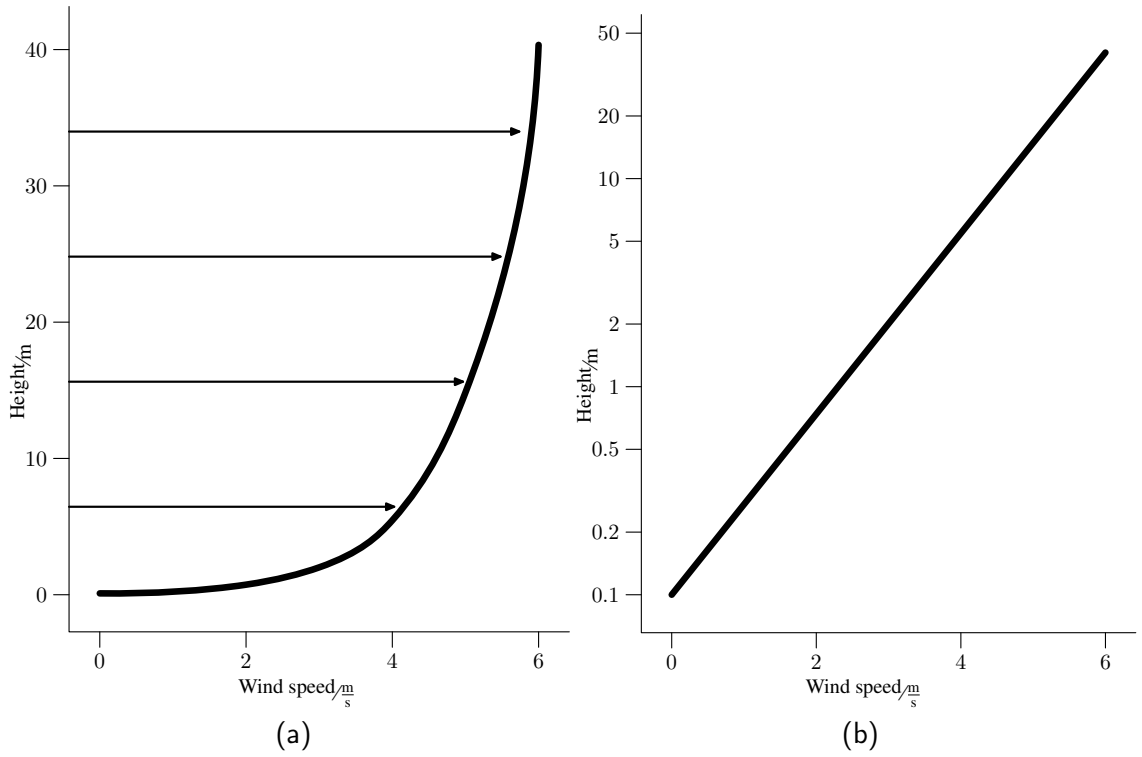


Figure 4.3: Logarithmic wind profile. Atmospheric boundary layer theory predicts a logarithmic increase of wind speed with height 4.3a. This yields a straight line plot when a logarithmic scale is used for height 4.3b.

complexity. The goal is to improve accuracy to a point at which shipboard landings are possible.

4.2 Wind Estimation and Profile Modeling for Shipboard Landing

Section 1.1.4 listed some of the existing challenges for landing vehicles onto the decks of ships. This section will explore additional issues that arise from the shipboard landing scenario that are specific to wind estimation and profile modeling.

4.2.1 Considerations for Shipboard Landing

The first main difference between traditional ADS employment to a land target and shipboard landing to be considered is the landing target's height above the earth's surface. In

the case of shipboard landing, the height of interest is the height of the ship's landing deck above sea level. When the logarithmic wind profile model as described in Section 4.1.4 is used, the *no slip* condition stipulates that the horizontal wind velocity at the sea surface should be zero. More precisely, the wind velocity becomes zero at a height above the sea surface equal to the aerodynamic roughness length (which is a very small value over the ocean's surface). For the shipboard landing case, the ADS should still compute the logarithmic wind profile down to the sea surface; however, the wind velocity at the landing height will be the one that the guidance algorithm will use to compute the landing trajectory.

The second consideration is the actual value of the wind at the landing height relative to the ship's deck, which is moving. In the context of shipboard helicopter operations, this value is called wind-over-deck (WOD). For landing helicopters, the ship maneuvers so that the WOD comes from either the port or starboard bow. In this way, the helicopter is able to maintain a higher airspeed, and thus lift on the rotor blades, while having a lower speed with respect to the landing deck. For a landing ADS, it is likely, in the case of a cooperative target, that the ship would maneuver so that the parafoil were flying downwind to the landing target. In this way, the ADS would not have to struggle upwind to land, and its downwind speed relative to the ship's deck could be controlled using the ship's throttle.

The third challenge inherent to landing on a moving platform at sea is turbulent airflow caused by the ship's superstructure. Characterization and visualization of the *air wake* of a ship underway as been the subject of some research, particularly for the application of helicopter operations; the article by Lee contains a survey [105]. These studies often rely on computational fluid dynamics (CFD) in order to predict the flow field around the ship's superstructure; however, an ongoing research project at the U.S. Naval Academy is correlating CFD results with flow field data gathered *in situ* on a 33 m training vessel that has been modified to include a scale model flight deck aft of the superstructure [106]–[108]. Figure 4.4 shows one example of correlation between CFD and *in situ* data, where black vectors represent the *in situ* data, and white vectors and the background color scale represent the CFD flow field. One prominent feature of Figure 4.4 is the region of recirculating flow immediately aft of the superstructure. This region of disturbed flow, sometimes called a *burble* by pilots, presents a challenge to the helicopter pilot upon landing.

For a landing ADS, the downward airflow over the landing area immediately aft of the

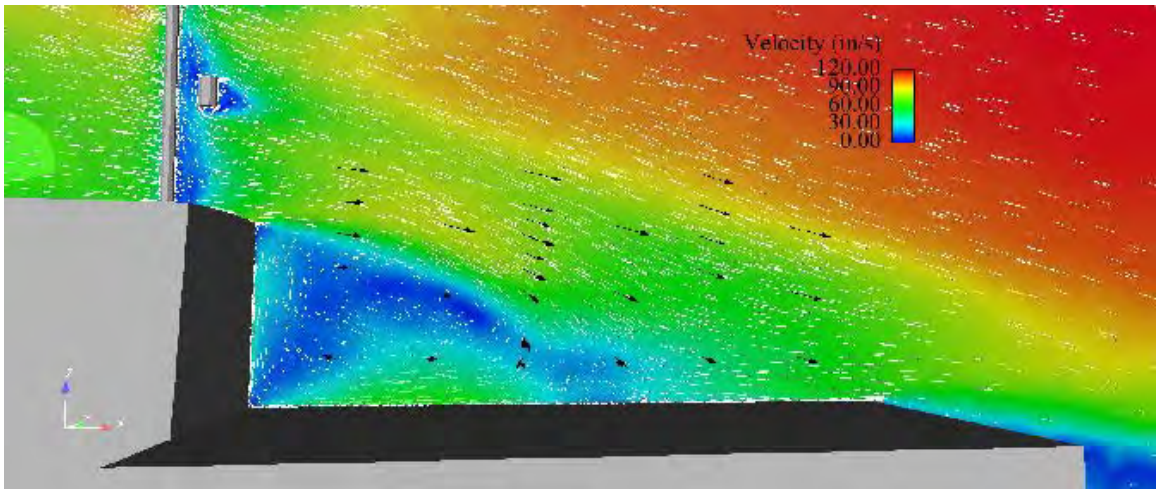


Figure 4.4: A research project at the U.S. Naval Academy has collected normalized *in situ* data (black vectors) along with 7 knots time-averaged CFD data (white vectors and color scale) concerning airflow aft of the superstructure (courtesy of M. Snyder [108]).

ship's superstructure will increase the vehicle's descent rate at the time of touchdown. Examining U.S. Naval Academy (USNA) experimental results published by Snyder, a downdraft velocity of 0.8 m/s appears to be approximately the average downdraft over the landing are for the ship making way at 7 knots. One technique that the ADS may employ to mitigate the effect of the burble is to add a small (1 m to 2 m) vertical offset to the target aim point. In effect, the ADS would be flying to a spot 1 m to 2 m above the center of the landing area.

4.2.2 Link between Profile Modeling and Trajectory Planning

Section 4.2.1 dealt with winds at a couple of specific points in the shipboard landing scenario. Additionally, the entire wind profile itself is a necessary input to an ADS guidance algorithm that strives for accurate landings. Implicit in this requirement is the need to determine exactly what the guidance algorithm will do with the wind profile information. Presumably, the ADS will alter its trajectory somehow based on the wind profile information; the AccuGlide and ParaLander systems mentioned in Chapter 2, Section 2.5.2 both initiate final maneuvers based on wind profile calculations [48], [49]. Even though these systems use simplistic wind models such as the constant wind profile to plan their trajectories, the fact that they do use this information demonstrates the linkage between wind modeling and trajectory planning. Therefore, a study of the use of wind models for an

ADS must be done in the context of a specific guidance algorithm. In the next section, the Snowflake ADS's guidance algorithm and its linkage to wind estimation and profile modeling will be examined in detail.

4.3 Description of the Snowflake ADS Guidance Algorithm

Understanding how an ADS's guidance algorithm uses wind information is fundamental to understanding the benefit of an accurate wind profile, as noted in Section 4.2.2. A detailed description of the Snowflake ADS guidance algorithm follows, which highlights two key decisions that must be made during flight involving the wind profile.

4.3.1 Aerial Delivery System Trajectory Planning Overview

Snowflake's guidance algorithm depends on a preflight wind estimate so that a desired release point relative to the ground target can be computed. The CARP represents the beginning point of Snowflake's flight trajectory. Due to the possibly overpowering effect of strong winds on the flight path of the ADS, the CARP is most often located upwind of the ground target so that the ADS has an easier downwind flight to the target. The Snowflake ADS plans its trajectory from the CARP to the target in two stages, an energy management, or *loitering* stage and an approach stage. An overview of different schemes for energy management can be found in a survey of autonomous parafoil GN&C algorithms in an article by Kaminer and Yakimenko [4].

During the loitering stage, the Snowflake ADS flies a holding pattern upwind of the target, performing wind estimation and calculating the moment when it should begin its approach. During the approach stage, the Snowflake flies downwind to a point offset from the ground target, then completes a 180° final approach turn, as detailed in the work by Slegers and Yakimenko [7]. The upwind landing enables a more accurate landing, and reduces the Snowflake's velocity relative to the ground upon impact. A diagram of the overview of the guidance strategy including the loitering pattern is shown in Figure 4.5. The loiter pattern is defined by four waypoints at the corners of the box pattern, labeled A, B, C, and D. The start of the final 180° turn is known as the Turn Initiation Point (TIP), and the distances *away* and *cycle* define, respectively, the proximity of the loiter pattern to the target, and the major dimension of the loiter pattern.

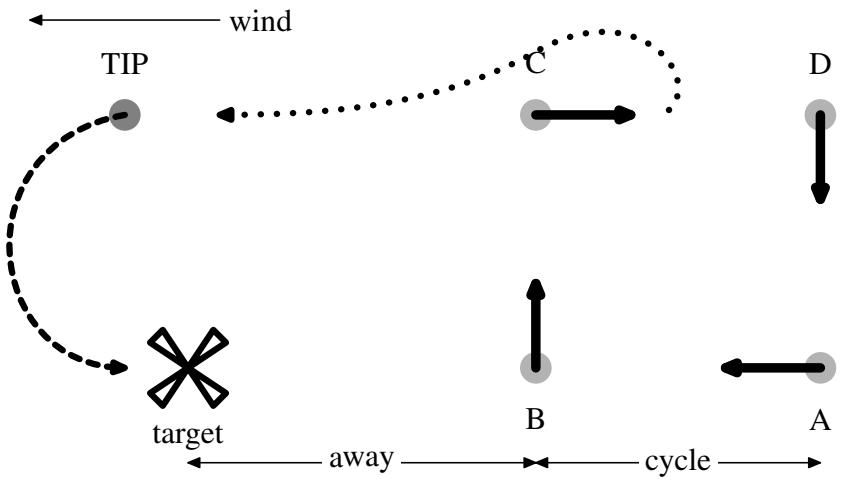


Figure 4.5: Snowflake ADS Guidance strategy overview. Depicted are the loiter pattern (solid arrows), loiter exit (dotted line), and the approach turn (dashed line).

For analysis and testing, the guidance strategy is divided into phases as summarized below [7].

Phase 0 After parafoil canopy first opens, no guidance commands are given for a few seconds while the Snowflake's flight path becomes steady.

Phase 1 Snowflake follows heading command to point A.

Phase 2 Snowflake executes waypoint navigation in the loiter pattern A–B–C–D.

Phase 3 Snowflake executes a turn out of the loiter pattern toward the TIP.

Phase 4 Snowflake lines up to fly directly downwind to the TIP.

Phase 5 Snowflake tracks the optimal turn trajectory towards the target.

Phase 6 Snowflake flies directly upwind along the target line until landing.

4.3.2 Wind Estimation in the Snowflake Guidance Algorithm

Any sophisticated ADS that uses a two-stage trajectory plan as described in Section 4.3.1 must decide when to switch from the loitering stage to the approach stage. For Snowflake, this decision is made based on a threshold altitude known as z_{start}, and it is the first of two key decisions that Snowflake makes during its approach. In Figure 4.5, the transition from the loitering stage to the approach stage is shown by the dotted-line flight path from the loiter pattern towards the TIP. Furthermore, Snowflake must make a second decision about when to begin the final 180° turn: either before or after it has reached abeam the

target. The distance from the abeam position to the TIP is given the label D_{switch}; a positive value indicates that the TIP is after the abeam position, and a negative value of D_{switch} indicates that the TIP is before the abeam position. The computation of where to locate the TIP is the second of the two key decisions that Snowflake makes during its approach. An overview diagram of the terminal guidance maneuver is shown in Figure 4.6. Note that the beginning of the flight path in this diagram labeled t_{start} corresponds to the altitude z_{start} at which Snowflake changed from the loitering stage to the approach stage of flight.

In summary, the two major decision milestones that the Snowflake ADS guidance algorithm must compute in flight are z_{start} and D_{switch}, in that order. The computation of D_{switch} is the final and most critical decision milestone and is the focus of Section 4.4.1.

Following the derivation by Slegers and Yakimenko [7], a three-dimensional, orthogonal coordinate frame is defined with its origin at the target, and axes as depicted in Figure 4.6. Note that in Figure 4.6, the representation of the coordinate axes is offset from the target symbol only to make the diagram less cluttered. The x axis is in the ground plane, and directed into the direction of the wind, with the wind direction assumed to be constant. The z axis is pointed down, perpendicular to the ground plane, and the y axis completes the triad according to the right-hand rule. One point about this convention is especially important with respect to the value of wind magnitude. If, for example, wind were blowing from the north, then the x axis would point north from the origin, and the magnitude of the wind would be assigned a positive value. Alternatively, if the wind were blowing from the south, then the x axis could still be directed north from the origin; but, in this case, the wind would be assigned a negative value. In fact, this second case is the one most often used when arranging for an upwind landing.

Expressions are then developed for the changes in Snowflake's position in the x and z directions, given known coordinates of the starting and ending positions. In the xz plane, the starting coordinates are $(-L, z_{\text{start}})$, with L being defined as a positive number, and z_{start} as a negative number. The ending coordinates at the target are $(0,0)$. Summing the starting value with the change in coordinate to equal the ending value both in x and in z, two equations are obtained, thus:

t_{start}	time of Snowflake commencing Phase 4 (downwind to TIP)
t_0	time of Snowflake arriving at TIP and commencing Phase 5
t_{exit}	time of Snowflake commencing Phase 6 (steer to target)
t_1	time of Snowflake intercepting final approach course
t_2	time of Snowflake touchdown
T_{downwind}	duration $t_0 - t_{\text{start}}$
T_{turn}	final turn duration $t_1 - t_0$
T_{app}	final approach duration $t_2 - t_1$
D_{switch}	optimal distance to pass the target before commencing final turn
$\tilde{D}$	displacement during final turn due to wind; positive value when point t_1 x coordinate less than that of t_0
L	distance from target line at t_{start}
L_{app}	straight approach distance from point t_1 to target
R	final turn radius
W	x component of wind velocity; positive value when blowing toward the origin
$\psi(t)$	reference raw function that is tracked in final turn

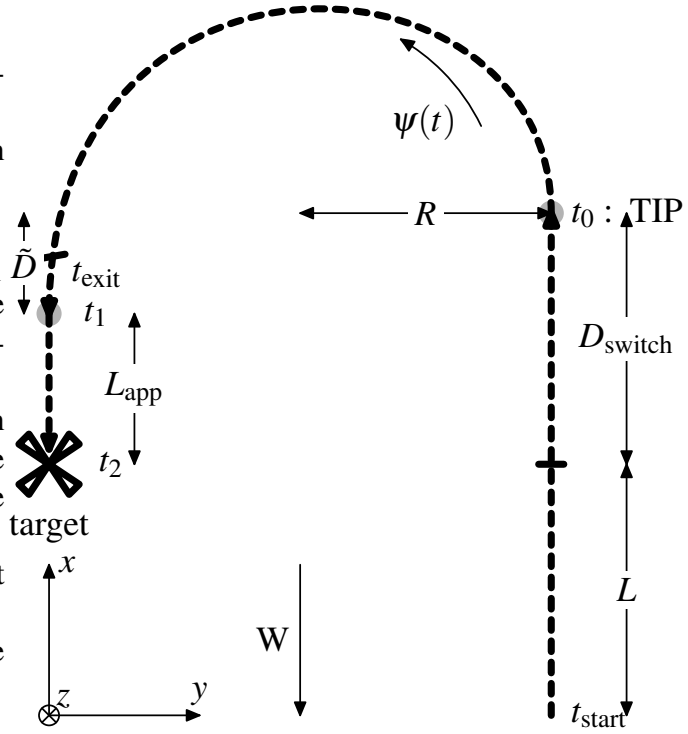


Figure 4.6: Snowflake ADS terminal guidance maneuver overview. The reference coordinate axes are actually centered on the target, but are shown offset here for clarity.

$$-L + L + D_{\text{switch}} + \underbrace{\int_{t_0}^{t_1} \dot{x}\, dt}_{-\tilde{D}} + \underbrace{\int_{t_1}^{t_2} \dot{x}\, dt}_{-L_{\text{app}}} = 0 \tag{4.2}$$

$$z_{\text{start}} + \int_{t_{\text{start}}}^{t_0} \dot{z}\, dt + \int_{t_0}^{t_1} \dot{z}\, dt + \int_{t_1}^{t_2} \dot{z}\, dt = 0. \tag{4.3}$$

Note that Equation (4.2) is the same equation as that presented in Ref. 7, which is the definitive description of the Snowflake guidance algorithm by Slegers and Yakimenko. Also, in Equation (4.3), z_{start} is a negative number; therefore, all $\dot{z}$ terms are positive (assuming the ADS only descends and never ascends).

In the work of Slegers and Yakimenko [7], a set of simplifying assumptions is made that enables explicit expressions to be derived for D_{switch} and z_{start}. One of these simplifying assumptions is that the wind profile, or the wind speed over a range of altitudes, is constant or piece-wise constant. The method used by the Snowflake ADS in flight to estimate wind speed is very simple and does not require the use of a pitot-static airspeed sensor. An *a priori* estimate of the wind direction is used to align the loiter pattern so that the long dimension of the rectangle *ABCD* is in the direction of the wind estimate. The result is that the two long segments of the rectangular loiter pattern should be aligned very nearly directly downwind and upwind. Referring to Figure 4.5, the *forward* velocity that will be measured by the on-board GPS receiver as the Snowflake travels in the direction of *A* to *B* on the downwind segment will be:

$$V_{\text{f}} = V_{\text{h}}^{\star} + W \tag{4.4}$$

and, similarly, the *reverse* velocity measured as the Snowflake travels in the direction of *C* to *D* on the upwind segment will be:

$$V_{\text{r}} = V_{\text{h}}^{\star} - W. \tag{4.5}$$

Using Equation (4.4) and Equation (4.5), including speed measurements from both the downwind and upwind segments, an estimate of the wind $\widehat{W}$ can be computed with:

$$\widehat{W} = \frac{V_{\text{f}} - V_{\text{r}}}{2}. \tag{4.6}$$

In practice, for each execution of the Snowflake main program loop, an estimate of the wind is calculated using Equation (4.6) and the most recently measured values of the velocities V_{f} and V_{r}.

4.4 Computing D_{switch} using a Logarithmic Wind Profile

Now that the wind profile models have been discussed in Section 4.2, and a specific example of an ADS guidance algorithm has been described in Section 4.3, this section will focus on using the logarithmic wind profile to calculate the important parameter D_{switch} that is used in the Snowflake guidance algorithm. There are two important assumptions necessary for the development in the rest of this section. The first assumption is that the shape of the wind profile is known. The logarithmic wind profile is defined by two parameters α and β as will be discussed in Section 4.4.1. The second assumption is that there is one constant prevailing wind direction throughout the entire vertical extent of the wind profile.

It is a well-established part of atmospheric boundary layer theory as presented by Stull [13] that the prevailing horizontal wind direction does vary with height due to the Coriolis effect and friction with the earth's surface; however, accounting for this effect causes the estimation problem to be very much more complex. Therefore, only wind profiles in one dimension will be considered. With these two assumptions in place, the geometry of Figure 4.6 again applies to this problem, with the difference that wind magnitude $W(z)$ now varies with height.

4.4.1 Iterative Calculation for D_{switch}

In the following derivation, the assumption of a constant or piece-wise constant wind profile will be replaced by the assumption that wind speed W varies logarithmically with altitude z as:

$$W(z) = \alpha \ln(-z) + \beta \tag{4.7}$$

where α and β are constants. Note that, since altitude z is represented as a negative number according to the sign convention in use, an additional factor of -1 is included in the argument of the logarithm. The derivation uses this assumption for the wind profile, along with the assumptions that the steady-state, no-wind values of the ADS horizontal and ver-

tical velocities are known and labeled $V_h^\star$ and $V_v^\star$. Furthermore, it is assumed that the wind vector lies only in the horizontal plane, and is parallel with the x axis of the coordinate system depicted in Figure 4.6. Using the assumptions of constant steady-state, no-wind vertical velocity, and no vertical wind component, the time derivative of the z coordinate, $\dot{z}$, is just equal to $V_v^\star$, where downward change has a positive sign. Thus, the integral terms in Equation (4.3) can be simplified to:

$$z_{\text{start}} + V_v^\star T_{\text{downwind}} + V_v^\star T_{\text{turn}} + V_v^\star T_{\text{app}} = 0 \tag{4.8}$$

where T_{downwind} represents the time duration required to travel the sum of distances L and D_{switch}. Again, recall that under the sign convention being used here, z_{start} is a negative number and that $V_v^\star$ and all time durations are positive. Assuming that the Snowflake's turn rate is constant, and that the turn will encompass 180° of a circle of known radius R, the time duration for the turn T_{turn} is a known constant in terms of R and $V_h^\star$. Therefore, this equation has only two unknowns: T_{downwind} and T_{app}.

Under the assumption of a constant wind profile, a simple expression for T_{downwind} could be written in terms of unknown value D_{switch}. Also using the constant wind profile assumption, Equation (4.2) could be simplified into an equation containing two unknowns: D_{switch} and T_{app}; therefore, the two linear equations in two unknowns could easily be solved. Under the assumption of a logarithmic wind profile as in Equation (4.7), T_{downwind} cannot be replaced by a simple expression containing D_{switch}. Instead, an iterative computation will be described that uses both Equations (4.2) and (4.8) to solve for D_{switch}. An initial value for D_{switch} will be chosen that will be used to calculate several intermediate parameters. These intermediate parameters are used in turn to calculate an updated value of D_{switch} that is then used as the starting value for the next iteration. The iteration is complete when the difference between successive updated values of D_{switch} falls below a predetermined threshold value. The sequence of computations is illustrated graphically in Figure 4.7. In the following subsections, derivations will be presented for the individual computations shown in Figure 4.7.

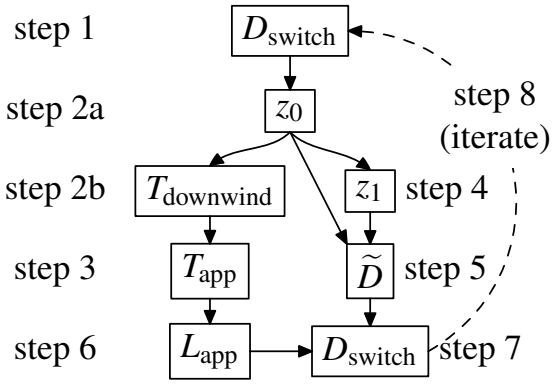


Figure 4.7: Sequence of iterative computations for D_{switch}. The steps in the figure refer to the computation summary in Section 4.4.5.

4.4.2 Derivation of Altitude at TIP z_0

Referring to Figure 4.6, z_0 is the ADS's altitude at the TIP. Note that the symbol z_0 here, and in the following sections, is used to refer to a particular Snowflake altitude instead of aerodynamic roughness length. As the ADS flies toward this point, it is affected by wind that changes with altitude. The computation of z_0 starts with the fundamental expression for movement along the *a priori* assumed wind direction between times t_{start} and t_0. The distance traveled, x, is resolved into a component that is due to no-wind velocity, and a component due to the wind. The resulting expression contains an unknown value of time, called t_0, instead of the corresponding altitude at that time, z_0. There exists a simple variable transformation from time to altitude based on the assumption of constant descent rate, as follows:

$$\int_{t_{\text{start}}}^{t_0} \dot{x}\, dt = L + D_{\text{switch}} \tag{4.9}$$

$$V_{\text{h}}^{\star}(t_0 - t_{\text{start}}) - \int_{t_{\text{start}}}^{t_0} W(z)\, dt = L + D_{\text{switch}}. \tag{4.10}$$

A variable transformation is performed on both terms, converting from time to altitude as the independent variable. Altitudes corresponding to t_{start} and t_0 are labeled z_{start} and z_0 respectively, thus:

$$z(t) = V_{\text{v}}^{\star}(t - t_{\text{start}}) + z_{\text{start}} \tag{4.11}$$

where the change in altitude is expressed in terms of time using the simple expression:

$$dz = V_{\mathrm{v}}^{\star} dt. \tag{4.12}$$

Using the relationship in Equation (4.12) in Equation (4.10) yields:

$$\frac{V_{\mathrm{h}}^{\star}}{V_{\mathrm{v}}^{\star}}(z_0 - z_{\mathrm{start}}) - \frac{1}{V_{\mathrm{v}}^{\star}} \int_{z_{\mathrm{start}}}^{z_0} W(z)\, dz = L + D_{\mathrm{switch}}. \tag{4.13}$$

Moving known quantity involving z_{start} to the right side, and substituting in Equation (4.7) for logarithmic wind profile, yields:

$$\frac{V_{\mathrm{h}}^{\star}}{V_{\mathrm{v}}^{\star}} z_0 - \frac{1}{V_{\mathrm{v}}^{\star}} \int_{z_{\mathrm{start}}}^{z_0} \alpha \ln(-z) + \beta \; dz = L + D_{\mathrm{switch}} + \frac{V_{\mathrm{h}}^{\star}}{V_{\mathrm{v}}^{\star}} z_{\mathrm{start}}. \tag{4.14}$$

Next, a variable substitution is performed to account for the fact that altitude is represented as a negative number in the sign convention in use:

$$\begin{aligned} z' &= -z \\ dz' &= -dz \end{aligned} \tag{4.15}$$

$$\frac{V_{\mathrm{h}}^{\star}}{V_{\mathrm{v}}^{\star}} z_0 + \frac{1}{V_{\mathrm{v}}^{\star}} \int_{-z_{\mathrm{start}}}^{-z_0} \alpha \ln z' + \beta \; dz' = L + D_{\mathrm{switch}} + \frac{V_{\mathrm{h}}^{\star}}{V_{\mathrm{v}}^{\star}} z_{\mathrm{start}} \tag{4.16}$$

and the definite integral is then evaluated as follows:

$$\frac{V_{\mathrm{h}}^{\star}}{V_{\mathrm{v}}^{\star}} z_0 + \frac{\beta}{V_{\mathrm{v}}^{\star}}(-z_0 + z_{\mathrm{start}}) + \frac{\alpha}{V_{\mathrm{v}}^{\star}} \Big[z' \ln z' - z' \Big]_{-z_{\mathrm{start}}}^{-z_0} = L + D_{\mathrm{switch}} + \frac{V_{\mathrm{h}}^{\star}}{V_{\mathrm{v}}^{\star}} z_{\mathrm{start}}. \tag{4.17}$$

Evaluating the limits of integration, and moving all known quantities to the right side,

assuming D_{switch} is a known quantity, results in the following expressions:

$$\left(\frac{V_{\text{h}}^{\star}-\beta}{V_{\text{v}}^{\star}}\right) z_0 + \frac{\alpha}{V_{\text{v}}^{\star}}\left[-z_0\ln(-z_0) + z_0 + z_{\text{start}}\ln(-z_{\text{start}}) - z_{\text{start}}\right] = L + D_{\text{switch}} + \left(\frac{V_{\text{h}}^{\star}-\beta}{V_{\text{v}}^{\star}}\right) z_{\text{start}} \tag{4.18}$$

$$\underbrace{\left(\frac{V_{\text{h}}^{\star}+\alpha-\beta}{V_{\text{v}}^{\star}}\right)}_{a} z_0 - \underbrace{\frac{\alpha}{V_{\text{v}}^{\star}}}_{b} z_0\ln(-z_0) = \underbrace{L + D_{\text{switch}} + \left(\frac{V_{\text{h}}^{\star}+\alpha-\beta}{V_{\text{v}}^{\star}}\right) z_{\text{start}} - \frac{\alpha}{V_{\text{v}}^{\star}} z_{\text{start}}\ln(-z_{\text{start}})}_{c}. \tag{4.19}$$

Equation Equation (4.19) is an equation in only one unknown, z_0, and has the form

$$az_0 + bz_0\ln(-z_0) = c. \tag{4.20}$$

The Lambert W function is described in detail in the work by Corless [109], and it can be used to solve equations of the form of Equation (4.20). The Lambert W function is defined as the function that satisfies:

$$W(z)e^{W(z)} = z. \tag{4.21}$$

Note here that the symbol $W(z)$ is traditionally used in the literature to represent the Lambert W function and its argument z. Elsewhere in this chapter, $W(z)$ can also refer to wind magnitude W at altitude z. The context of the usage should indicate which meaning the symbol represents.

Equation (4.20) can be manipulated into a form to which the Lambert W can be applied:

$$\ln(-z_0) + \frac{a}{b} = \frac{c}{bz_0} \tag{4.22}$$

$$-\frac{ce^{a/b}}{b} = \frac{c}{bz_0}e^{\frac{c}{bz_0}} \tag{4.23}$$

$$W\left(-\frac{ce^{a/b}}{b}\right) = \frac{c}{bz_0}. \tag{4.24}$$

The solution for altitude z_0 in terms of the Lambert W function is:

$$z_0 = \frac{c}{bW\left(-\frac{ce^{a/b}}{b}\right)}. \tag{4.25}$$

Constants a, b, and c are can be expressed in terms of known quantities $V_{\mathrm{h}}^{\star}$, $V_{\mathrm{v}}^{\star}$, α, β, L, z_{start}, and assumed quantity D_{switch}.

4.4.3 Derivation of Displacement During Final Turn $\widetilde{D}$

The computation of the displacement due to wind during the final turn, $\widetilde{D}$, starts with an expression for movement in the x direction between times t_0 and t_1. The sign convention established by Slegers and Yakimenko [7] is that the distance $\widetilde{D}$ is measured in the negative x direction from point t_1. In other words, if point t_1 has an x coordinate that is less than that of point t_0, then distance $\widetilde{D}$ is said to have a positive value. Put in yet another way, if, during the turn from point t_0 to point t_1, the wind pushes Snowflake closer to the target, then $\widetilde{D}$ is said to have a positive value; conversely, if the wind pushes Snowflake further away from the target during the turn, then $\widetilde{D}$ is said to have a negative value. Therefore, $\widetilde{D}$ has the opposite sign of the change in x coordinate between times t_0 and t_1:

$$\widetilde{D} = -\int_{t_0}^{t_1} \dot{x}\, dt. \tag{4.26}$$

For the next step, it is assumed that the turn trajectory from t_0 to t_1 is planned in such a way that, if there were no wind, that point t_1 would have the same x coordinate value as point t_0. Therefore, any deviation, measured as $\widetilde{D}$, is a result of the wind alone. Noting that the sign convention for wind in Figure 4.6 is such that positive values of wind speed W correspond to negative values of resultant movement in the x direction, or $\dot{x}$, Equation (4.26) is restated as:

$$\widetilde{D} = \int_{t_0}^{t_1} W(z)\, dt. \tag{4.27}$$

A variable transformation is performed, as in Equation (4.12), on the integral term, converting from time to altitude as the independent variable. Altitudes corresponding to t_0 and

t_1 are labeled z_0 and z_1, respectively:

$$\widetilde{D} = \frac{1}{V_{\mathrm{v}}^{\star}} \int_{z_0}^{z_1} W(z)\, dz. \tag{4.28}$$

The logarithmic wind profile in Equation (4.7) is then applied to yield:

$$\widetilde{D} = \frac{1}{V_{\mathrm{v}}^{\star}} \int_{z_0}^{z_1} \alpha \ln(-z) + \beta\, dz. \tag{4.29}$$

A variable substitution as in Equation (4.15) is performed to account for the fact that altitude is represented as a negative number in the sign convention in use:

$$\widetilde{D} = -\frac{1}{V_{\mathrm{v}}^{\star}} \int_{-z_0}^{-z_1} \alpha \ln z' + \beta\, dz'. \tag{4.30}$$

The definite integral and the limits of integration are then evaluated, and terms are grouped:

$$\widetilde{D} = \frac{\beta}{V_{\mathrm{v}}^{\star}}(z_1 - z_0) - \frac{\alpha}{V_{\mathrm{v}}^{\star}} \Big[z' \ln z' - z' \Big]_{-z_0}^{-z_1} \tag{4.31}$$

$$\widetilde{D} = \frac{\beta}{V_{\mathrm{v}}^{\star}}(z_1 - z_0) - \frac{\alpha}{V_{\mathrm{v}}^{\star}} \Big[-z_1 \ln(-z_1) + z_1 + z_0 \ln(-z_0) - z_0 \Big] \tag{4.32}$$

$$\widetilde{D} = \frac{\beta}{V_{\mathrm{v}}^{\star}}(z_1 - z_0) - \frac{\alpha}{V_{\mathrm{v}}^{\star}}(z_1 - z_0) - \frac{\alpha}{V_{\mathrm{v}}^{\star}} \Big[z_0 \ln(-z_0) - z_1 \ln(-z_1) \Big]. \tag{4.33}$$

Altitude is again related to time using Equation (4.11), yielding the following relations:

$$z(t_1) \equiv z_1 = V_{\mathrm{v}}^{\star}(t_1 - t_{\mathrm{start}}) + z_{\mathrm{start}} \tag{4.34}$$

$$z_1 - z_0 = V_{\mathrm{v}}^{\star}(t_1 - t_0) \equiv V_{\mathrm{v}}^{\star} T_{\mathrm{turn}}. \tag{4.35}$$

Using Equation (4.35), a final expression is obtained that includes values that are known *a priori*: α, β, T_{turn}, and $V_{\mathrm{v}}^{\star}$. This expression also includes altitude values z_0 and z_1 at the beginning and the end of the turning maneuver. Once one of these altitude values is determined or estimated, the other is easily found using the known time for the turning

maneuver, T_{turn} and the assumed constant descent rate $V_{\text{v}}^{\star}$:

$$\widetilde{D} = (\beta - \alpha) T_{\text{turn}} - \frac{\alpha}{V_{\text{v}}^{\star}} \Big[z_0 \ln(-z_0) - z_1 \ln(-z_1) \Big]. \tag{4.36}$$

4.4.4 Derivation of Approach Distance L_{app}

The distance L_{app} of the last, straight-in approach leg, is computed starting with the fundamental expression for movement in the x direction between times t_1 and t_2. Time t_2 is the time of landing. The distance L_{app} is defined as a positive distance; however, the change in x coordinate between times t_1 and t_2 is negative. Therefore, L_{app} will have the opposite sign from the integral of $\dot{x}$:

$$L_{\text{app}} = -1 \cdot \int_{t_1}^{t_2} \dot{x} \, dt. \tag{4.37}$$

The motion is then resolved into a component that is due to no-wind velocity, and a component due to the wind. During this segment of the flight path, the no-wind velocity $V_{\text{h}}^{\star}$ and motion due to positive values of wind W are both assumed to be aligned in the opposite direction of positive values of $\dot{x}$; therefore a negative sign is inserted into the integrand:

$$L_{\text{app}} = -1 \cdot \int_{t_1}^{t_2} -(V_{\text{h}}^{\star} + W(z)) \, dt. \tag{4.38}$$

Next, the logarithmic wind profile in Equation (4.7) is applied, and the times t_1 and t_2 are transformed into their corresponding altitudes, which are labeled z_1 and z_2 respectively, to yield:

$$L_{\text{app}} = \frac{1}{V_{\text{v}}^{\star}} \int_{z_1}^{z_2} V_{\text{h}}^{\star} + W(z) \, dz. \tag{4.39}$$

At this point, an additional feature of the wind model must be incorporated. The value of the wind magnitude $W(z)$ is assumed not to change sign with changing altitude; in other words, the output values of the function $W(z)$ should be either all positive or all negative over the range of altitude from z_{start} to the surface. The complication is that at values of altitude z close to zero, the sign of the natural logarithm function does change sign. Therefore, for this wind model, the value of the wind magnitude $W(z)$ shall be considered to be zero for

values of altitude z close to zero. It can be verified that for $z = -e^{-\beta/\alpha}$, the value of wind magnitude $W(z) = \alpha \ln(-z) + \beta = 0$; therefore, $W(z) = 0$ for $|z| < |e^{-\beta/\alpha}|$. The integral in Equation (4.39) can then be separated into two components: one with a non-zero value of $W(z)$ for $-e^{-\beta/\alpha} > z > z_1$ and one with a zero value for $W(z)$ for $0 > z > -e^{-\beta/\alpha}$. Note that z_2 is the touchdown point, so that its altitude is zero:

$$L_{\text{app}} = \frac{1}{V_{\text{v}}^{\star}} \left\{ \int_{z_1}^{-e^{-\beta/\alpha}} V_{\text{h}}^{\star} + \alpha \ln(-z) + \beta \; dz \right\} + \frac{1}{V_{\text{v}}^{\star}} \int_{-e^{-\beta/\alpha}}^{0} V_{\text{h}}^{\star} \; dz. \tag{4.40}$$

For both integrals, the variable substitution is performed as in Equation (4.15) to account for the fact that altitude is represented as a negative number in the sign convention in use:

$$L_{\text{app}} = \frac{-1}{V_{\text{v}}^{\star}} \left\{ \int_{-z_1}^{e^{-\beta/\alpha}} V_{\text{h}}^{\star} + \alpha \ln z' + \beta \; dz' \right\} - \frac{1}{V_{\text{v}}^{\star}} \int_{e^{-\beta/\alpha}}^{0} V_{\text{h}}^{\star} \; dz'. \tag{4.41}$$

The definite integrals and the limits of integration are evaluated, and terms are grouped in the following manner:

$$L_{\text{app}} = \frac{V_{\text{h}}^{\star}}{V_{\text{v}}^{\star}} \left(-e^{-\beta/\alpha} - z_1 + e^{-\beta/\alpha} \right) - \frac{\beta}{V_{\text{v}}^{\star}} \left(e^{-\beta/\alpha} + z_1 \right) - \frac{\alpha}{V_{\text{v}}^{\star}} \left[z' \ln z' - z' \right]_{-z_1}^{e^{-\beta/\alpha}} \tag{4.42}$$

$$L_{\text{app}} = \left(\frac{-V_{\text{h}}^{\star} - \beta}{V_{\text{v}}^{\star}} \right) z_1 - \frac{\beta e^{-\beta/\alpha}}{V_{\text{v}}^{\star}} - \frac{\alpha}{V_{\text{v}}^{\star}} \left[e^{-\beta/\alpha} \ln e^{-\beta/\alpha} - e^{-\beta/\alpha} + z_1 \ln(-z_1) - z_1 \right] \tag{4.43}$$

$$L_{\text{app}} = \left(\frac{-V_{\text{h}}^{\star} + \alpha - \beta}{V_{\text{v}}^{\star}} \right) z_1 + \frac{\alpha e^{-\beta/\alpha}}{V_{\text{v}}^{\star}} - \frac{\alpha}{V_{\text{v}}^{\star}} \left[z_1 \ln(-z_1) \right]. \tag{4.44}$$

Using the assumption of constant descent rate $V_{\text{v}}^{\star}$, the altitude z_1 at the beginning of the approach segment is just the time duration of the approach segment multiplied by the vertical velocity, expressed as a negative number:

$$z_1 = -T_{\text{app}} V_{\text{v}}^{\star}. \tag{4.45}$$

Substituting Equation (4.45) into Equation (4.44), a final expression is obtained in terms of values α, β, $V_{\mathrm{v}}^{\star}$, and $V_{\mathrm{h}}^{\star}$ that are known *a priori*, and value T_{app} that is easily computed using the known value T_{turn} and an estimated value for T_{downwind}:

$$L_{\mathrm{app}} = (V_{\mathrm{h}}^{\star} - \alpha + \beta)T_{\mathrm{app}} + \frac{\alpha e^{-\beta/\alpha}}{V_{\mathrm{v}}^{\star}} + \alpha T_{\mathrm{app}} \ln(T_{\mathrm{app}} V_{\mathrm{v}}^{\star}). \tag{4.46}$$

4.4.5 Summary of Iterative Calculation for D_{switch}

The iteration procedure to calculate D_{switch} is summarized below:

1. Select an initial guess value for D_{switch}.
2. Using the initial guess value for D_{switch}:
   (a) Calculate z_0 using the Lambert W function as shown in Equation (4.25).
   (b) Calculate T_{downwind} from:

$$T_{\mathrm{downwind}} = \frac{z_0 - z_{\mathrm{start}}}{V_{\mathrm{v}}^{\star}}. \tag{4.47}$$

3. Using T_{downwind}, calculate T_{app} using Equation (4.8):

$$T_{\mathrm{app}} = \frac{-z_{\mathrm{start}}}{V_{\mathrm{v}}^{\star}} - T_{\mathrm{turn}} - T_{\mathrm{downwind}}. \tag{4.48}$$

4. Compute z_1 using the assumption of constant vertical velocity:

$$z_1 = z_0 + V_{\mathrm{v}}^{\star} T_{\mathrm{turn}}. \tag{4.49}$$

5. Compute $\widetilde{D}$ from known quantities:

$$\widetilde{D} = (\beta - \alpha)T_{\mathrm{turn}} - \frac{\alpha}{V_{\mathrm{v}}^{\star}}\left[-z_0 \ln(-z_0) - z_1 \ln(-z_1)\right]. \tag{4.50}$$

6. Compute L_{app} using T_{app} using:

$$L_{\mathrm{app}} = (V_{\mathrm{h}}^{\star} - \alpha + \beta)T_{\mathrm{app}} + \frac{\alpha e^{-\beta/\alpha}}{V_{\mathrm{v}}^{\star}} + \alpha T_{\mathrm{app}} \ln(T_{\mathrm{app}} V_{\mathrm{v}}^{\star}). \tag{4.51}$$

7. Compute D_{switch} from:

$$D_{\text{switch}} = \widetilde{D} + L_{\text{app}}. \tag{4.52}$$

8. Use the value of D_{switch} computed in step 7 as the next iteration estimate of D_{switch} to be used in step 1. Repeat the iterations until differences between subsequent values of D_{switch} are below a specified threshold value.

4.5 Adaptive Filtering for Parameter Estimation

In Section 4.4.1, the parameters of the logarithmic wind profile, α and β, were treated as known parameters. These two parameters are estimated from the output of an adaptive filtering algorithm, which is shown in block diagram form in Figure 4.8. In this representation, the logarithmic wind model is interpreted in a discrete-time form with desired signal $d[n]$ containing the sequential estimates of wind speed from the Snowflake guidance algorithm. These sequential estimates are obtained using Equation (4.6) as described in Section 4.3.2. When converting the logarithmic wind profile model in Equation (4.7) into discrete-time form for the adaptive filtering algorithm, then measurement matrix $\mathbf{H}[n]$ contains the natural logarithm of the current altitude measurement $z[n]$. The estimate of wind speed $\hat{d}[n]$ at any altitude $z[n]$ is given by the following product:

$$\hat{d}[n] = \underbrace{\begin{bmatrix} \ln(-z[n]) & 1 \end{bmatrix}}_{\mathbf{H}[n]} \underbrace{\begin{bmatrix} \alpha \\ \beta \end{bmatrix}}_{\hat{\Theta}[n]}. \tag{4.53}$$

The term on the right side of Equation (4.53) can be represented as the product of the measurement matrix $\mathbf{H}[n]$ (a row vector in this case) and an estimated parameter vector $\hat{\Theta}[n]$. Note that the *true* parameter vector $\Theta[n]$ remains unknown. In the adaptive filtering model, the true output (desired signal $d[n]$) is known; however, the *true* logarithmic wind profile parameters and the measurement noise signal $v[n]$ that compose the true wind speed are unknown.

The RLS adaptive filtering algorithm can be applied to this parameter estimation problem; various digital signal processing textbooks contain the derivation of this algorithm [110], [111]. In summary, this algorithm minimizes the total error given by the following sum of

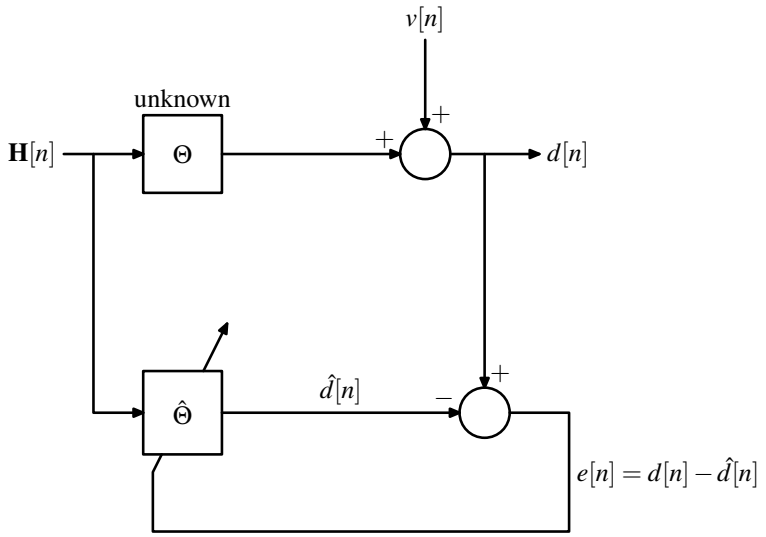


Figure 4.8: RLS adaptive filtering algorithm block diagram. Estimated parameter vector $\hat{\Theta}$ is adjusted to make the filter output $\hat{d}[n]$ match the desired signal $d[n]$ as closely as possible.

weighted error squares:

$$\Xi[n] = \sum_{i=1}^{n} \lambda^{n-i} |e[i]|^2 = \sum_{i=1}^{n} \lambda^{n-i} \left| d[i] - H[i]\hat{\Theta}[n] \right|^2. \tag{4.54}$$

where λ is known as the *forgetting factor* and typically has a value equal to or greater than 0.9 [110]. For the experiments described in this chapter, λ was set to 0.9. Therefore, the few most recent values of the *a posteriori* error signal $e[n]$ are the most influential in the calculation of total error $\Xi[n]$. As each new value of the measurement matrix $\mathbf{H}[n]$ and the desired output signal $d[n]$ is obtained, the new values of the estimated parameters $\hat{\Theta}[n]$ are calculated by adding to the previous estimate $\hat{\Theta}[n-1]$ a quantity proportional to the *a priori* error value $\xi[n]$. The *a priori* error is formed by using the previous parameter estimate values $\hat{\Theta}[n-1]$ with current measurement matrix $\mathbf{H}[n]$ and subtracting the product from the current desired signal value $d[n]$. The constant of proportionality in this operation is known as the adaptation gain vector $\mathbf{K}[n]$, which depends in turn on the measurement matrix $\mathbf{H}[n]$ and a 2×2 matrix (for the current problem) known as the *inverse correlation matrix* $\mathbf{P}[n]$. The RLS adaptive filtering algorithm is summarized as follows:

1. Initialize the inverse correlation matrix $\mathbf{P}(0)$ using the identity matrix $\mathbf{I}$ and a small

positive constant δ thus:

$$\mathbf{P}(0) = \delta\mathbf{I}. \tag{4.55}$$

2. Compute the adaptation gain vector $\mathbf{K}[n]$ by

$$\mathbf{K}[n] = \frac{\lambda^{-1}\mathbf{P}[n-1]\mathbf{H}^{\mathrm{T}}[n]}{1+\lambda^{-1}\mathbf{H}[n]\mathbf{P}[n-1]\mathbf{H}^{\mathrm{T}}[n]}. \tag{4.56}$$

3. Compute the *a priori* error value $\xi[n]$ using:

$$\xi[n] = y[n] - \mathbf{H}[n]\hat{\Theta}[n-1]. \tag{4.57}$$

4. Update the estimate of the parameters of the wind model $\hat{\Theta}[n]$:

$$\hat{\Theta}[n] = \hat{\Theta}[n-1] + \mathbf{K}[n]\xi[n]. \tag{4.58}$$

5. Update the value of the inverse correlation matrix $\mathbf{P}[n]$ using the relation:

$$\mathbf{P}[n] = \lambda^{-1}\mathbf{P}[n-1] - \lambda^{-1}\mathbf{K}[n]\mathbf{H}[n]\mathbf{P}[n-1]. \tag{4.59}$$

6. Increment discrete counter n, obtain new values of desired signal $d[n]$, and altitude $z[n]$ (which is used in forming the one-dimensional measurement matrix $\mathbf{H}[n]$) and then repeat starting at step 2.

4.6 Wind Estimation Algorithm Implementation

Up to this point, a method for Snowflake to use a sequence of altitude and velocity measurements in order to estimate the parameters of a logarithmic wind profile has been proposed in Section 4.5, and a means to use those parameters in the calculation of decision milestone D_{switch} has been described in Section 4.4. In the sequel, the incorporation of these new methods into the existing code for the Snowflake ADS guidance algorithm will be described, along with the implications of these changes.

Section 4.3.2 listed the two major decision milestones to be computed in flight as transition altitude z_{start} and transition distance D_{switch}. As also mentioned in that section, of the two decision milestones, D_{switch} was the final and more critical of the two. In the following

algorithm, the estimated parameters of the logarithmic wind profile are applied to the calculation of only D_{switch} for the additional reason that the transition altitude z_{start} is typically at an altitude above the limit of where the surface layer logarithmic wind profile assumption is considered to be valid (approximately up to a maximum of 200 m) [13]. Therefore, it is assumed that the z_{start} computation, for the altitude at which to exit the loitering phase, is made according to the derivation in Slegers and Yakimenko [7] using a constant wind profile assumption.

4.6.1 Trajectory Planning to a Moving Target

Kinematic equations were introduced in Chapter 1 for the calculation of z_{start} when the target is moving with constant velocity V_{T} (Equation (1.9) in Section 1.1.2). In this section, the goal is to outline methods for using the procedure for calculating D_{switch} derived in Section 4.4 in the case of a moving target.

One method for computing D_{switch} for a moving target is to do all calculations with respect to a set of coordinate axes that are attached to the target itself. Because it is assumed that the target is moving with a constant speed and direction during the time of flight of the ADS (see Chapter 3), this *target-fixed* frame is also in inertial reference frame. To use the target-fixed frame, the velocity of the ADS in the kinematic equations needs to be replaced with the relative velocity between the ADS and the target.

Another method for computing D_{switch} for a moving target is to use a fixed global reference frame. This can be done using the assumption of constant speed and direction of the target by computing the global coordinates at which the target will be when the ADS reaches the surface. These coordinates, called the *intercept point*, can then be used in the same manner as in the case of a stationary target.

When using the kinematic equations, the time remaining until intercept can be computed using either a relative or a global reference frame; the two methods are mathematically equivalent in this sense.

In Section 4.6.2, the advantages of the relative coordinate frame method will be examined in greater detail. Notwithstanding these advantages, in Section 4.6.3 and beyond in this chapter, the intercept point method using global coordinates will be used. The reason for

the use of global coordinates in the simulations to be described in Section 4.7 is that the optimal final turn algorithm used in the simulation (detailed in Ref. 7) was formulated for a fixed target on land, and that part of the simulation was not changed.

4.6.2 Relative Coordinates for Trajectory Planning

The use of relative coordinates for planning the landing trajectory helps with two separate issues at once. The first issue concerns the optimal final turn algorithm just mentioned. The optimal turn algorithm implemented using IDVD as detailed in the article by Slegers and Yakimenko [7] strives to have the ADS at a particular position and orientation (at the target and heading upwind) at a particular time (the moment of touchdown). Using relative coordinates, the final stage of the navigation algorithm must enable the ADS to arrive at the final location at *or before* the time of touchdown.

Trajectories that bring the ADS to the final location *before* touchdown are effective if the relative motion between the ADS and the target is suitable. Consider the case in which the target is moving in a straight line with a constant speed equal to the ADS's forward velocity when it is in the landing phase. In a coordinate frame fixed to the target, the position of the ADS is stationary when its forward speed matches that of the target. Figure 4.9 shows successive positions of an ADS relative to the target. The target ship's course is south, and the ADS begins its trajectory on the ship's port bow, and also flying south while completing its final turn to a course of due south. The wind is from due south, so both ship and ADS are traveling upwind. As the ADS descends, its ground speed increases to match that of the target due to decreasing head winds; therefore, as shown in Figure 4.9, the successive lower (darker) parafoil positions are not approaching the ship as quickly. If the final trajectory is planned in relative coordinates to reach a stationary point above the landing area, then the ADS and the target can continue moving together with the same velocity until the ADS touches down.

A trajectory planned using relative coordinates also mitigates the problem of air wake turbulence from the ship's superstructure as the ADS is landing. During the landing phase, the ADS and target are both moving with zero relative velocity between them. Therefore, any downward flow affecting the ADS will simply force the ADS to land on the deck sooner.

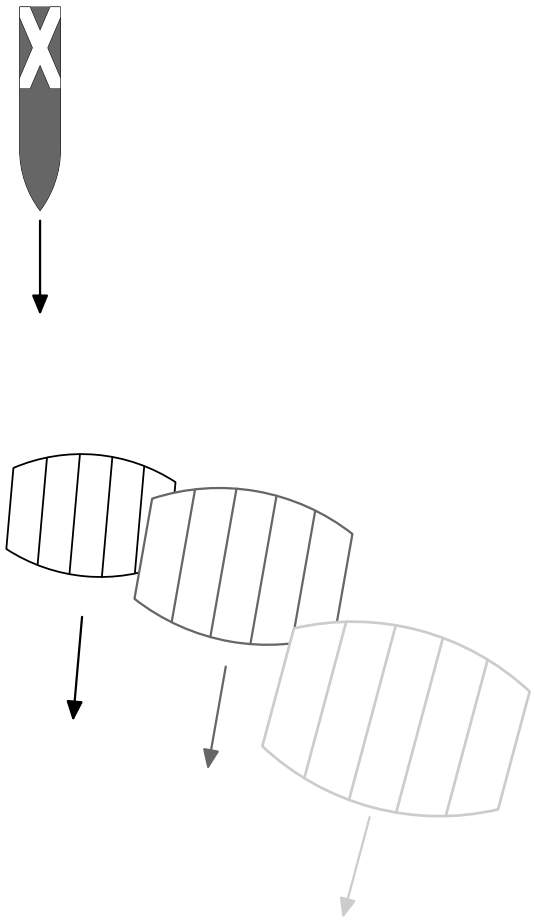


Figure 4.9: Landing motion of the ADS in a relative coordinate frame. This diagram shows the positions over time of the landing ADS relative to the ship target as the ADS descends and while completing its final turn.

4.6.3 Logarithmic Wind Profile with Snowflake Guidance Algorithm

The incorporation of the algorithm into the Snowflake ADS code would be as follows:

1. At each execution of the main loop, measure current altitude $z[n]$ and, if on a downwind or upwind segment, measure V_{f} or V_{r} as appropriate.
2. If altitude is below a threshold value, for instance 200 m AGL, and Snowflake is in phase 4, use adaptive filter algorithm as described in Section 4.5 in order to estimate parameters α and β of logarithmic wind profile. If not yet in phase 4, return to step 1.
3. Use current values of parameters α and β in iterative computation as described in Section 4.4.1 to compute D_{switch}.

4. Compare Snowflake's current x coordinate (as defined in Figure 4.6). If $x < D_{\text{switch}}$, return to step 1.
5. If $x > D_{\text{switch}}$, then begin phase 5 and start final turn to target. Note that the comparison of x to D_{switch} is valid whether D_{switch} is negative or positive.

It should be seen in simulation that a Snowflake relying on the logarithmic wind profile should, all other aspects being equal, begin the final turn towards the target later than a Snowflake relying on the assumption of a constant wind profile because the logarithmic wind profile should predict smaller values of wind speed near the surface.

4.7 Dynamic Parafoil Model Simulation

To test the assumption just stated about the later final turn for an algorithm using the logarithmic wind profile, a simulation was used that combined a dynamic model of an ADS along with a wind profile based on recorded dropsonde data. The dynamic model used was one that calculated the equations of motion for the parafoil and payload, which together comprise the ADS, in six degrees of freedom (6DOF). These six degrees consist of a translation of the entire system in three dimensions, and a rotation of the entire system about three orthogonal axes. In a 6DOF simulation, the payload and canopy together move and rotate as a rigid body. For this work, two models were readily available: one written in MATLAB by Slegers for the small Snowflake prototype ADS [7], and one by Mortaloni and Yakimenko written in Simulink for the larger Pegasus 500 ADS [112]. For the simulation in this section, the MATLAB model by Slegers was used. This model has also been the basis of other work such as that by Ward, Montalvo, and Costello [51].

The recorded wind profiles were from a set of flight tests conducted at YPG in 2008 by Yakimenko, Slegers, and Tiaden [11]. The profiles were derived from the data from dropsondes, which were introduced in Section 2.5.2. The data from the dropsondes indicated that the prevailing wind direction did change with height; however, as discussed in Section 4.4, the wind profile modeling algorithm assumes a constant direction. Therefore, the simulation was programmed with the Snowflake ADS's one-dimensional wind estimation algorithm described in Section 4.3.2 to estimate wind along this direction.

4.7.1 6DOF Simulation Varied Parameters

Monte Carlo studies have been run before using this 6DOF model simulation. Slegers and Yakimenko have run many instances of this 6DOF model while varying simulated measurement noise in the ADS's position and attitude sensors [7]. As part of the same study, they also varied the direction and magnitude of a linear wind profile. The varied wind profiles used in those simulations were varied in angle away from the algorithm's *a priori* direction; however, each profile contained horizontal wind vectors that were all co-planar at the various altitudes. The recorded wind profiles for the simulation described in this section featured different wind directions and different wind magnitudes at each altitude.

Slegers and Yakimenko determined that the sources of error they introduced in their simulation resulted in landing errors that were mostly in the downrange direction, not the cross-range direction [7]. For this current study, the 6DOF simulation was used to determine whether a set of varied parameters effected cross-range or downrange error. This study was not done in the Monte Carlo style, in which the parameters would be varied stochastically; rather, one deterministic instance of the simulation was run for each set of parameter values.

For this set of simulations, the primary varied parameter was the wind profile model used: constant or logarithmic. Landing error was then resolved into downrange and cross-range directions. Two other parameters were also varied in this set of simulations: desired approach time and the cross-wind component of the simulated winds. For the moving target scenario, it was thought that a longer straight-in approach to the target would be more effective; so, cases with a 20 s final approach, and a 60 s final approach were run. For the cross-wind component, there were two cases: one with the existing cross-wind component from the recorded wind profile, and one case with the cross-wind component set to zero. The cross-wind component was eliminated by setting the directions of all recorded wind vectors to the *a priori* direction.

One very important issue involving the desired final approach time is that it is used for the calculation of the loiter exit altitude z_{start}, and that this computation *always* uses the constant wind profile assumption. Furthermore, the estimated final approach time in flight, T_{app}, is also *always* computed using the constant wind profile assumption. A desired final approach time of 20 s or 60 s will result in a higher computed value of z_{start} which will subsequently magnify the landing errors due to an early turn to final. The algorithm using the logarithmic wind profile compensates for the possibly inaccurate computed values of z_{start} and T_{app} by using a more accurate D_{switch} calculation.

In these simulations, the wind is from the south and the ADS flies downwind to the target. The target holds a constant course and speed upwind toward the ADS. The first comparison set run consisted of no cross-wind and a short (20 s) final approach time. This first comparison is described next.

4.7.2 One-dimensional Wind Profile Results

In the 6DOF simulation, the simulated Snowflake has uncorrupted measurements of its own position and ground-speed velocity with which to estimate the wind field through which it is moving. A plot of the Snowflake horizontal wind magnitude estimates from a direction of due south versus height is shown in Figure 4.10. Also on this plot, the horizontal wind velocity measurements from the wind-pack are shown projected on to the direction of due south. These recorded wind-pack measurements are used as the true wind field for the simulation. In Figure 4.10, this true wind field is shown only from phase 4 onward. For the Snowflake estimates, it can be seen that the estimate is not being updated during phase 4. This is due to a feature of the Snowflake algorithm called the *wind timer* that causes the wind estimation algorithm to pause during and after the turn out of the loiter pattern. The delay is to allow the Snowflake to reach a steady course downwind toward the target. In this case, the duration of phase 4 is shorter than that of the wind timer; therefore, no wind velocity updates are computed during phase 4. Furthermore, the Snowflake algorithm in this 6DOF model does not perform wind estimation during phase 5, so the estimate remains constant during that phase also. The wind-pack's measured wind profile shown in Figure 4.10 does decrease significantly with decreasing height; this recorded profile was selected specifically for this characteristic. This choice was made to test the hypothesis in Section 4.6.3 that the algorithm using the logarithmic wind profile would execute a later final turn.

Figure 4.11 shows the simulated trajectory of the ADS for this run. The critical guidance algorithm decision for initiating the final turn occurs at the transition between phase 4 and phase 5. In Figure 4.11, this decision occurs before the Snowflake has reached the abeam position during phase 4. Therefore, the simulated Snowflake turns before reaching the abeam decision, and lands past the target, as expected. The $\times$ symbol represents the target location at the time of the ADS landing.

The other simulation run in this first pair used the logarithmic wind profile. Figure 4.12 shows the wind estimates made by the Snowflake until phase 5; thereafter the plot contains the shape of the logarithmic profile computed using parameters α and β. Compared to Figure 4.10, the Snowflake in this simulation entered phase 5 at a lower altitude—approximately 90 m versus 130 m in Figure 4.10. A later initiation of the phase 5 final

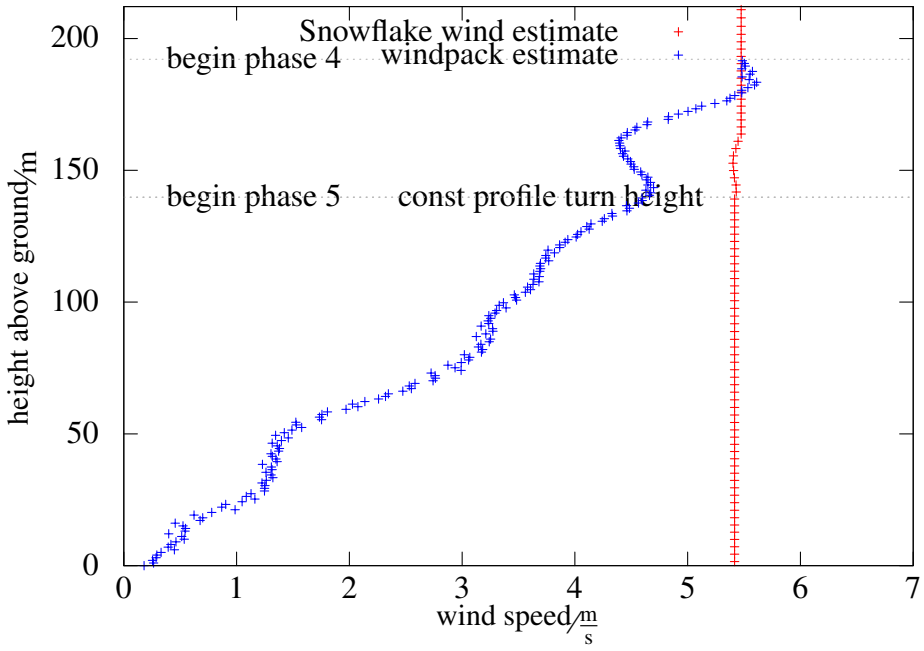


Figure 4.10: Constant wind profile in 6DOF simulation. Horizontal wind velocity estimates computed by the ADS are shown along with the true wind profile that was derived from windpack data. The true wind profile is shown only from phase 4 onward.

turn is represented in the trajectory view in Figure 4.13. Notice that the phase 5 trajectory in Figure 4.13 is the trajectory *flown*, not the trajectory planned.

A plot versus time of the computed values of D_{switch} along with the distance L_x to the abeam position allows a closer look at the critical final turn decision. Figure 4.14 is a plot of these values versus time. This plot indicates that the Snowflake's calculation of D_{switch}, done using the constant wind profile assumption, becomes negative and remains negative at approximately 160 s of simulation time, prior to phase 3. A negative value of D_{switch} represents a planned turn prior to the abeam position. During phases 3 and 4, the Snowflake is turning and flying downwind toward the target, so the distance L_x to the abeam position is decreasing. When L_x has decreased to be less than $-D_{\text{switch}}$, then the guidance algorithm initiates the final turn.

In Figure 4.15, the same plot is shown for the simulation run using the logarithmic wind profile. Until the beginning of phase 4, the guidance algorithm is computing D_{switch} using

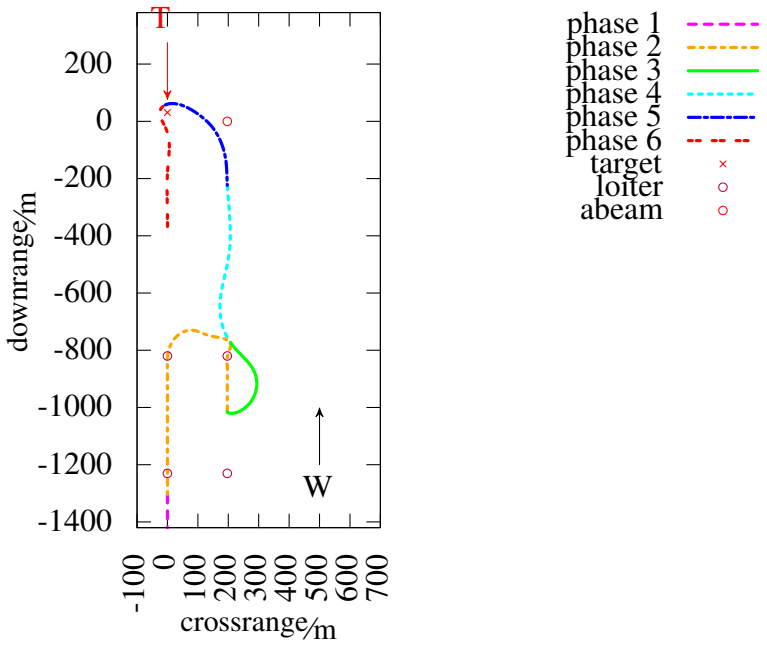


Figure 4.11: Constant wind profile simulation trajectory. The wind vector is denoted with a capital W and the target motion vector with a capital T. The $\times$ symbol represents the target location at the time of the ADS landing.

the constant wind profile; therefore, the plot in Figure 4.14 matches that in Figure 4.15 during the time up to the start of phase 4. In Figure 4.15, the guidance algorithm begins using the logarithmic wind profile after the beginning of phase 4; this instant is evident when the computed value of D_{switch} suddenly jumps above the dotted line drawn at $D_{\text{switch}} = 0$. A positive value for D_{switch} means that the ADS should fly past the abeam point; therefore, and ADS using the logarithmic wind profile will fly further before starting the final turn than one using the constant wind profile assumption.

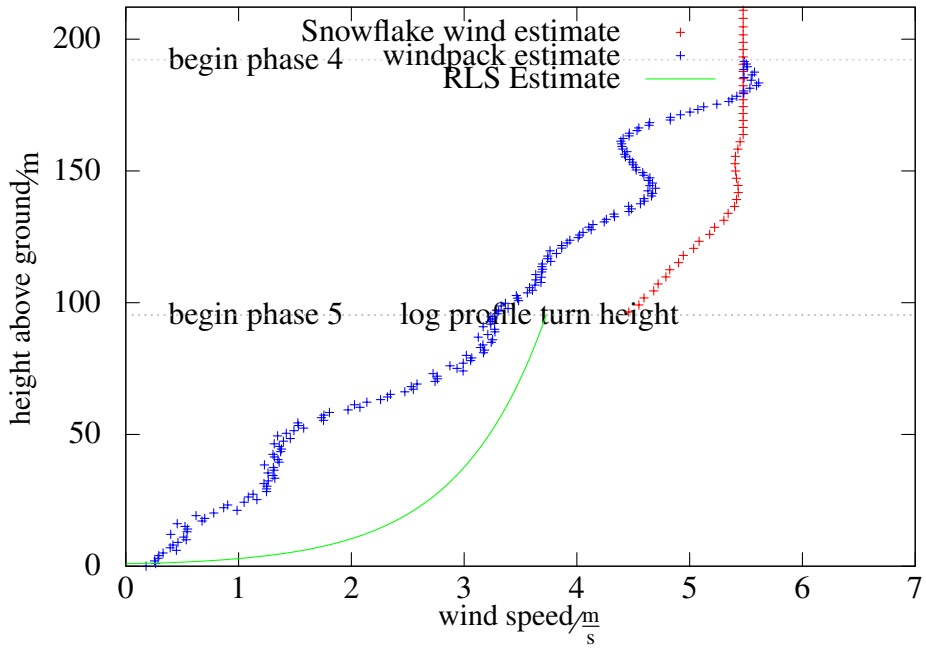


Figure 4.12: Logarithmic wind profile in 6DOF simulation. After the beginning of phase 5, the plot contains the shape of the logarithmic profile computed using parameters α and β.

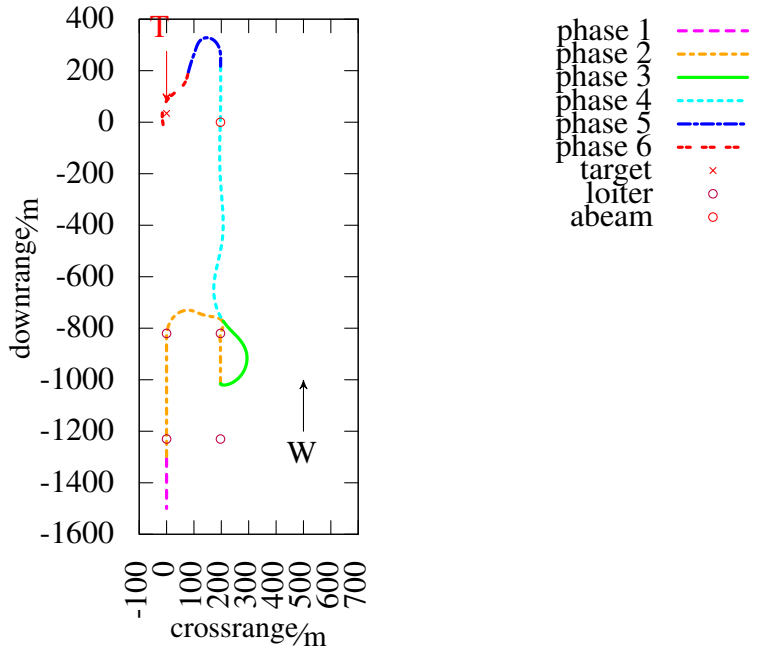


Figure 4.13: Logarithmic wind profile simulation trajectory. The wind vector is denoted with a capital W and the target motion vector with a capital T. The $\times$ symbol represents the target location at the time of the ADS landing.

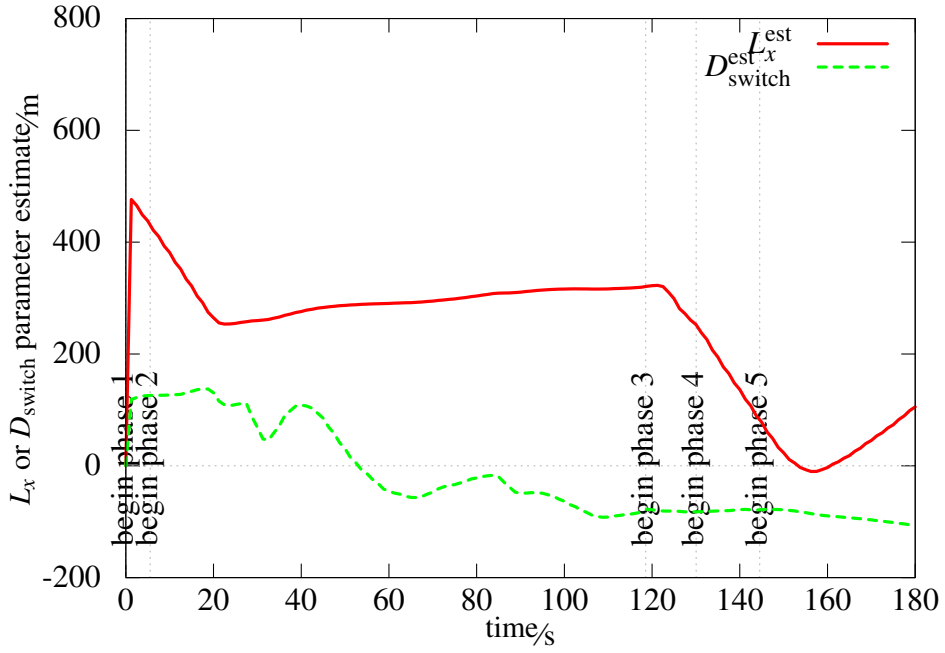


Figure 4.14: Final turn parameters L_x and D_{switch} plotted versus time. The guidance algorithm initiates the final turn (marking the start of phase 5) when $L_x < D_{\text{switch}}$.

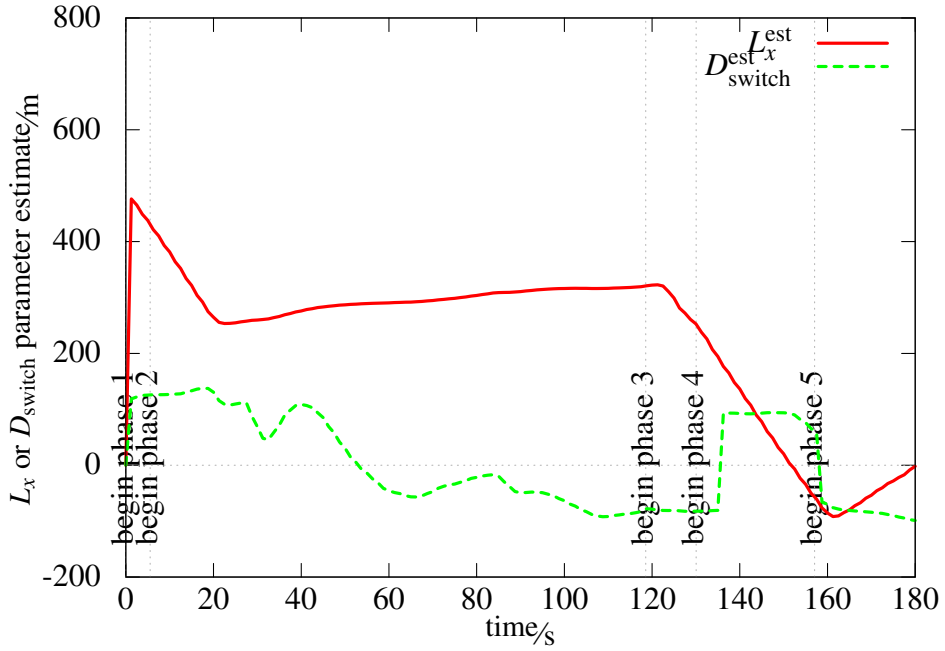


Figure 4.15: Final turn parameters L_x and D_{switch} plotted versus time. The guidance algorithm initiates the final turn (marking the start of phase 5) when $L_x < D_{\text{switch}}$.

4.7.3 Overall 6DOF Simulation Results

One simulation was run for each combination of the varied parameters discussed in Section 4.7.1, and the downwind and cross-wind components of the final landing error are shown in Table 4.1. The different performance of the algorithm using the constant wind profile versus one using a logarithmic wind profile was analyzed in detail in Section 4.7.2. Those results correspond to the case of no cross-wind and a short approach time in the upper-left quadrant of Table 4.1. For those simulations that were run with a cross-wind component, the recorded winds used as an input to the simulation varied from southeast to southwest. Therefore, the wind profile modeling algorithm was programmed to use due south as its *a priori* direction.

Table 4.1: Overall results of simulation trade study. Downwind and cross-wind components of landing error are shown for each combination of parameters.

		short approach		long approach	
simulation winds	profile model	downwind error/m	cross-wind error/m	downwind error/m	cross-wind error/m
no cross-wind	constant	107.4	0.1	150.3	0.0
	logarithmic	5.5	3.8	12.3	0.0
cross-wind	constant	115.1	0.1	152.7	0.0
	logarithmic	10.5	1.3	2.4	0.1

These results present more evidence that the use of a logarithmic wind profile can reduce significantly the downwind landing error, as previously shown in the trajectory plots in Figure 4.11 and Figure 4.13. Another observation is that the cross-wind landing error is in all cases smaller than the downwind landing error. The optimal final turn algorithm by Slegers and Yakimenko is designed to achieve this result [7]. When cross-wind error did occur, it occurred more often in runs with the shorter approach time of 20 s. The longer approaches gave the ADS more time to maintain the final approach track; however, these longer approaches also increased the effect of an imperfect D_{switch} calculation. The algorithm using the logarithmic wind profile produced smaller values of landing error using either the long or short landing approach in all cases compared to the algorithm using the constant wind profile.

The wind profile estimation plot and the trajectory plot are shown in Figures 4.16 and 4.17

for the case of no cross-wind, logarithmic wind profile, and long final approach. With the longer approach time, the ADS began the final turn sooner (at a higher altitude) as seen in Figure 4.16 compared to Figure 4.12. From the trajectory plot in Figure 4.17, it can be seen that the ADS is able to fly a straight final approach track with no cross-wind error.

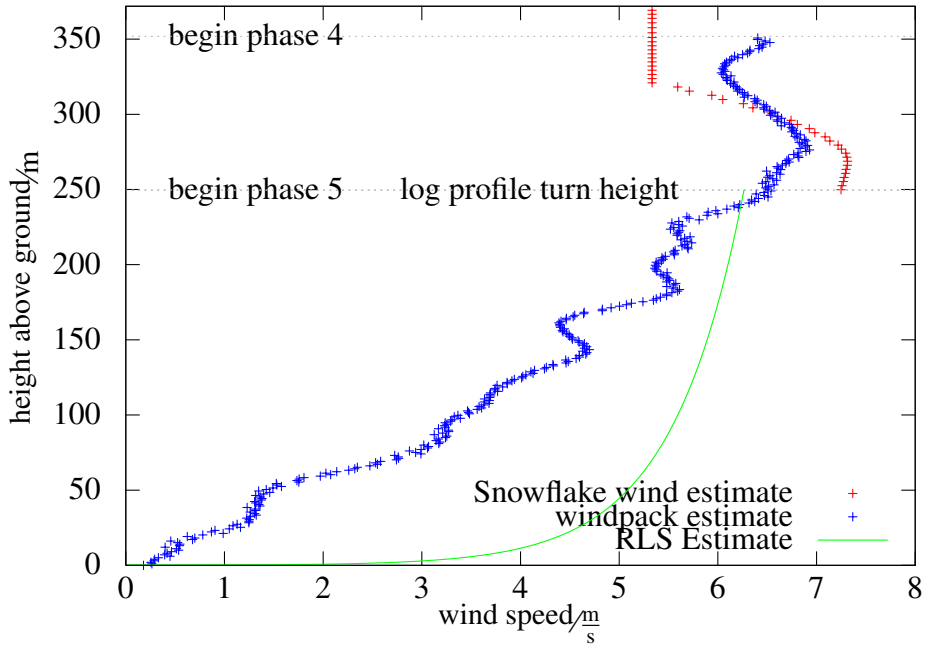


Figure 4.16: Logarithmic wind profile in 6DOF simulation. After the beginning of phase 5, the plot contains the shape of the logarithmic profile computed using parameters α and β.

4.8 Post-processing Flight Test Data

After the 6DOF simulations were analyzed, the next step was to include actual Snowflake recorded flight data in the simulation. The algorithm detailed in Section 4.6 for applying a logarithmic wind model to horizontal wind estimates generated in flight was first applied to recorded Snowflake telemetry data from previous flight tests.

4.8.1 Using Snowflake 1D Wind Estimates

The data set from flight tests at YPG, Yuma, Arizona, in October 2008 [11] was chosen for processing first due to the existence of radiosonde data concurrent with the Snowflake flights. The recorded Snowflake horizontal wind estimates were processed after the flight using the RLS algorithm in order to determine what the logarithmic wind model parameters

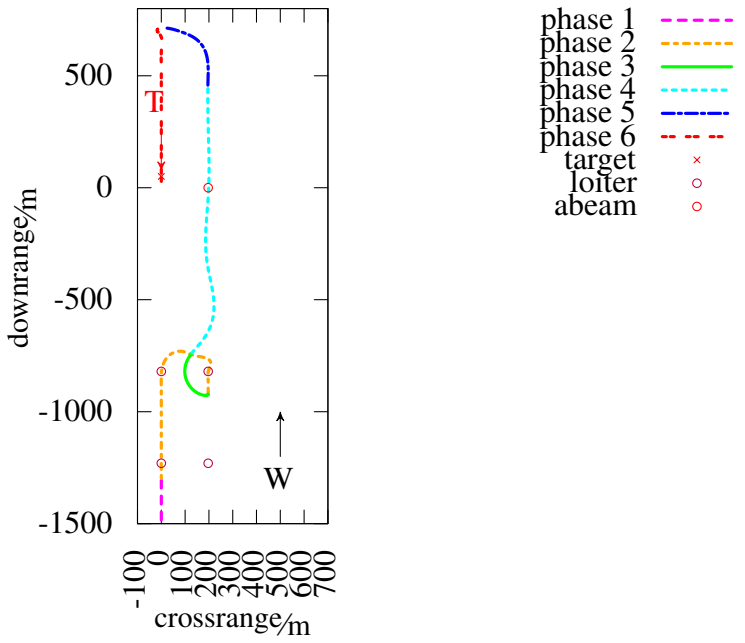


Figure 4.17: Logarithmic wind profile simulation trajectory. The wind vector is denoted with a capital W and the target motion vector with a capital T. The $\times$ symbol represents the target location at the time of the ADS landing.

α and β would have been had they been calculated in flight. Because Snowflake's recorded wind estimates are no longer updated once the Snowflake commences phase 5, the final turn to target, the last values of α and β calculated were used to generate a logarithmic wind profile that was then compared to the radiosonde data at altitudes corresponding to those of the Snowflake during final approach. Figure 4.18 illustrates such a comparison.

In Figure 4.18, data is presented from drop number 4 on 20 October 2008 from a test series conducted at YPG [11]. This particular drop happened to be the one with the largest miss distance of the two-day test series, and it was the subject of a detailed analysis by Yakimenko, Slegers, and Tiaden [11]. The original flown trajectory of this drop resulted in overshoot, as is depicted in Figure 4.19. A later start to the final turn maneuver would have reduced the miss distance of this drop.

The constant wind profile turn height depicted in Figure 4.18 was also computed by the post-processing script, and it is the height at which the Snowflake would have commenced the final turn after correction of a computation error in the 2008 version of the Snowflake's

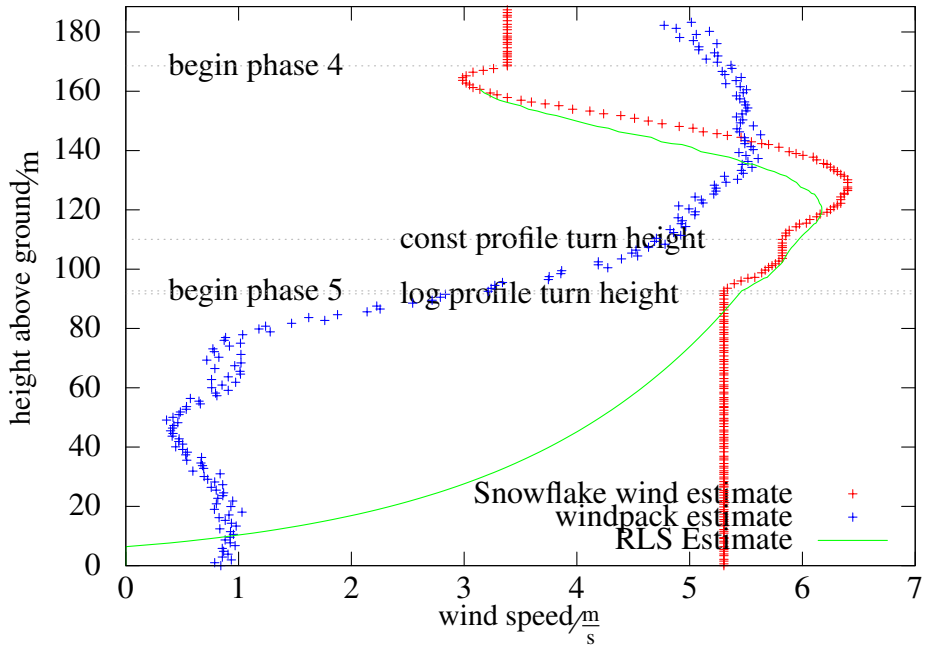


Figure 4.18: Constant wind profile compared to logarithmic in simulation. Flight test data from October 2008 from a guidance algorithm that used a constant wind profile is compared to computations using a logarithmic wind profile calculated in simulation. Computed final turn altitudes are shown for each algorithm.

flight software.[1] The corrected constant wind profile turn height of 110 m is shown in Figure 4.18.

The values of the RLS parameters α and β were used to calculate new values for D_{switch} using the logarithmic wind profile. The turn height that would correspond to this new value of D_{switch} is also shown as a horizontal line on Figure 4.18. The post-processed calculation of D_{switch} indicates that the Snowflake would have commenced the final turn later than was done using the constant wind assumption, which would have been a correct decision in this case.

4.8.2 Computing RLS Parameters in Flight

The next step in development was to have the RLS parameters computed in flight during flight tests conducted at Camp Roberts, California, 21 to 24 February 2011. These trials are

[1] In the 2008 version of the Snowflake's autopilot source code, D_{switch} was computed with the incorrect variable for steady-state forward velocity, which resulted in a later-than-normal turn to the final approach.

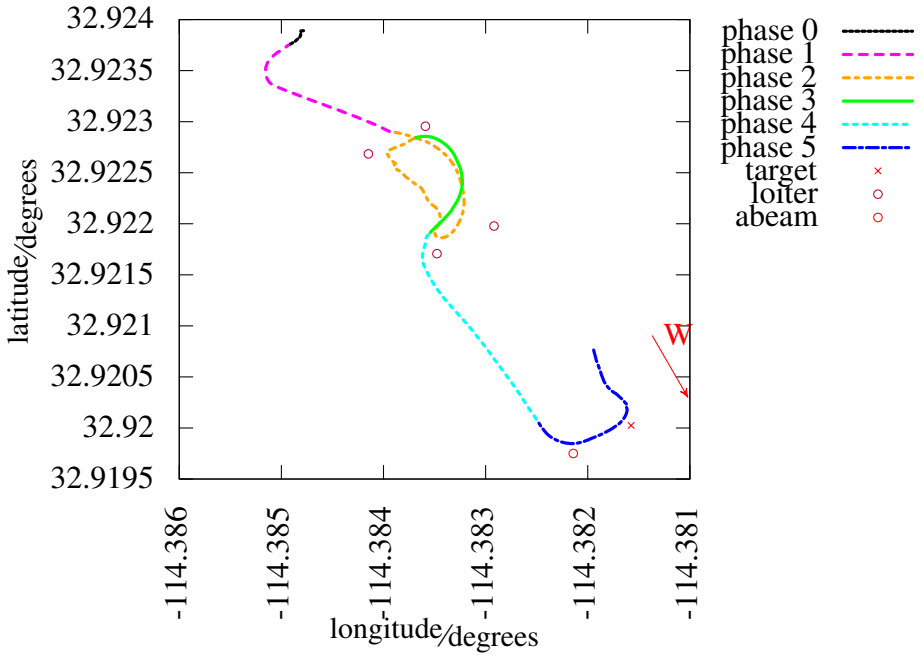


Figure 4.19: Parafoil trajectory overview using constant wind profile. This trajectory is from a flight test conducted in October 2008.

described as simulations, because, while the Snowflake was indeed generating horizontal wind speed estimates in flight and using those to calculate logarithmic wind model parameters α and β, the Snowflakes were not using this information to determine a value for D_{switch} based on the logarithmic wind profile assumption. Rather, the final turn decision for the Snowflakes in these trials was based upon the constant wind profile assumption, and the value of D_{switch} for the logarithmic wind profile was computed after the flight using a MATLAB script.

The estimated logarithmic wind profile generated in post-flight simulation using the wind estimates gathered in flight from drop number 6 on 24 February 2011 is shown in Figure 4.20. The value of D_{switch} computed using the logarithmic wind profile is compared to the actual turn start point computed assuming a constant wind profile, and the turn using the logarithmic wind profile would have occurred later.

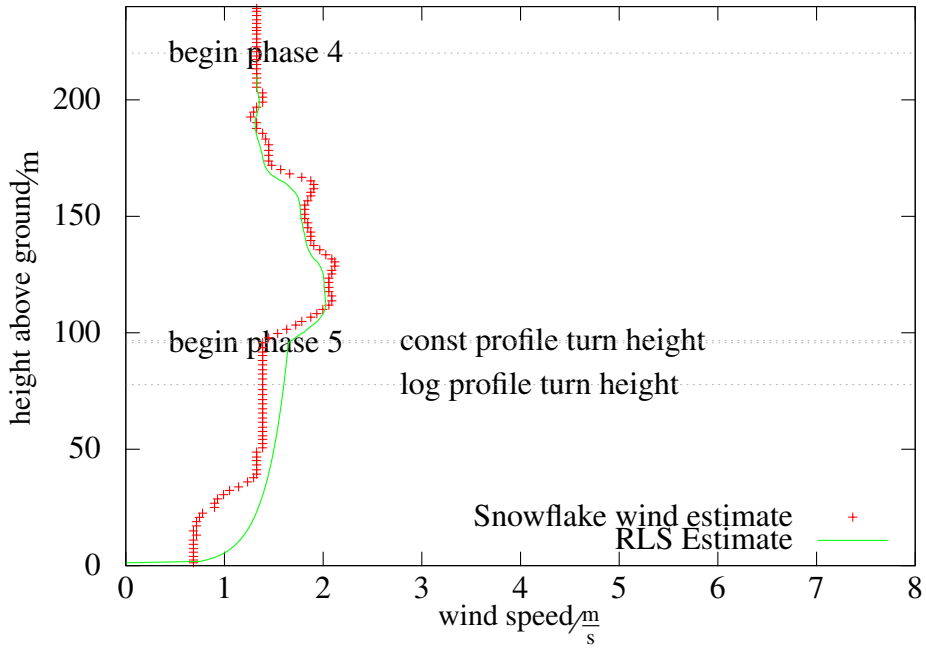


Figure 4.20: Constant wind profile compared to logarithmic in flight test. From a February 2011 flight test, a Snowflake ADS computed a constant wind profile in flight, and a logarithmic wind profile was calculated using afterward using RLS parameters determined in flight. Computed final turn altitudes are shown for each algorithm.

4.9 Flight Test Comparison of Wind Profiling Methods

The full implementation of the adaptive filtering algorithm as described in Section 4.6, including the calculation of D_{switch} was subsequently incorporated into the source code for the autopilot installed in the Snowflake ADS. The systems thus modified were tested in flight after being dropped from an altitude between 762 m to 914 m (2500 to 3000 ft) AGL by an Arcturus T-20 UAS over McMillan Airfield (identifier CA62) at Camp Roberts, California, in a series of tests during 2 to 4 May 2011.

The in-flight wind estimate calculated by the Snowflake from drop number 2 on 3 May 2011, using the logarithmic wind profile model is plotted in Figure 4.21. As the Snowflake begins phase 5 (final approach turn) and phase 6 (straight flight to target), the wind profile estimate produced by the final calculated values of α and β is shown along with the actual wind estimates produced during phase 6. From these two plots, it can be inferred that the logarithmic model agrees reasonably well with the measured data.

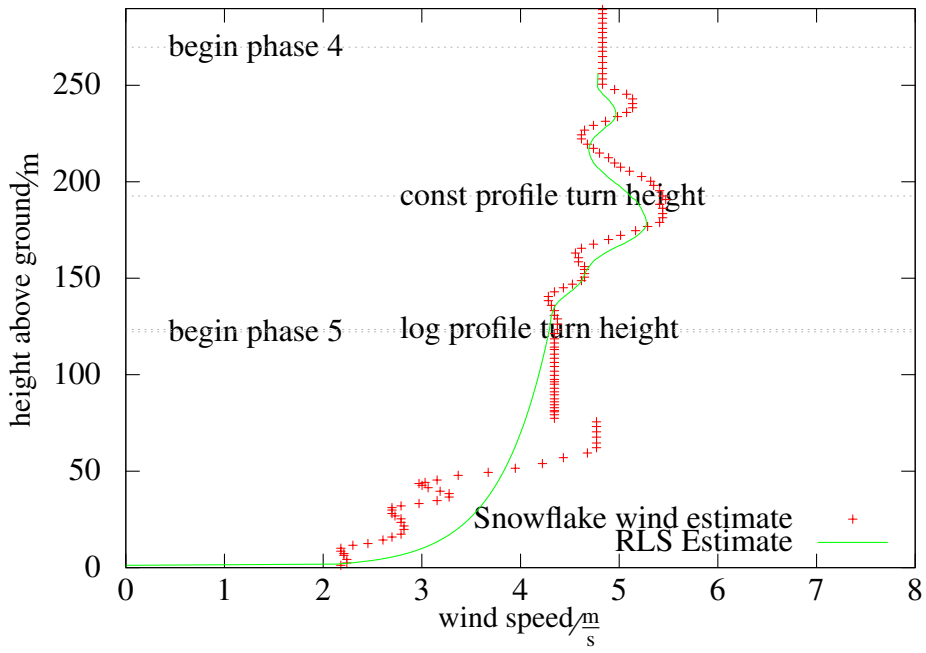


Figure 4.21: Logarithmic wind profile compared to constant profile in simulation. From a May 2011 flight test, the logarithmic wind profile was computed in flight, and a constant wind profile was calculated in a post-flight simulation using the in-flight wind measurements. Computed final turn altitudes are shown for each algorithm.

The estimates made by a Snowflake in flight of the two parameters L_x and D_{switch} are shown in Figure 4.22. Measured parameter L_x is similar to distance L illustrated in 4.6, and is defined the *current* distance along the x-axis from the Snowflake to a position abeam of the target. This value is updated during every iteration of the main guidance loop in the Snowflake software. This parameter is defined as positive when the Snowflake has not yet reached the abeam position, and negative after the Snowflake has flown past the abeam position.

Because D_{switch} is defined such that positive values represent a turn past the abeam position, and negative values indicate a turn prior to the abeam position, the criterion upon which the Snowflake algorithm commences the final turn is $L_x < -D_{\text{switch}}$. In Figure 4.22, it can be seen that at the start of phase 4, when the guidance algorithm began calculating D_{switch} using the logarithmic wind profile model, its D_{switch} estimate went from negative (turn prior to the target) to positive (turn after abeam the target). The third data series plotted in

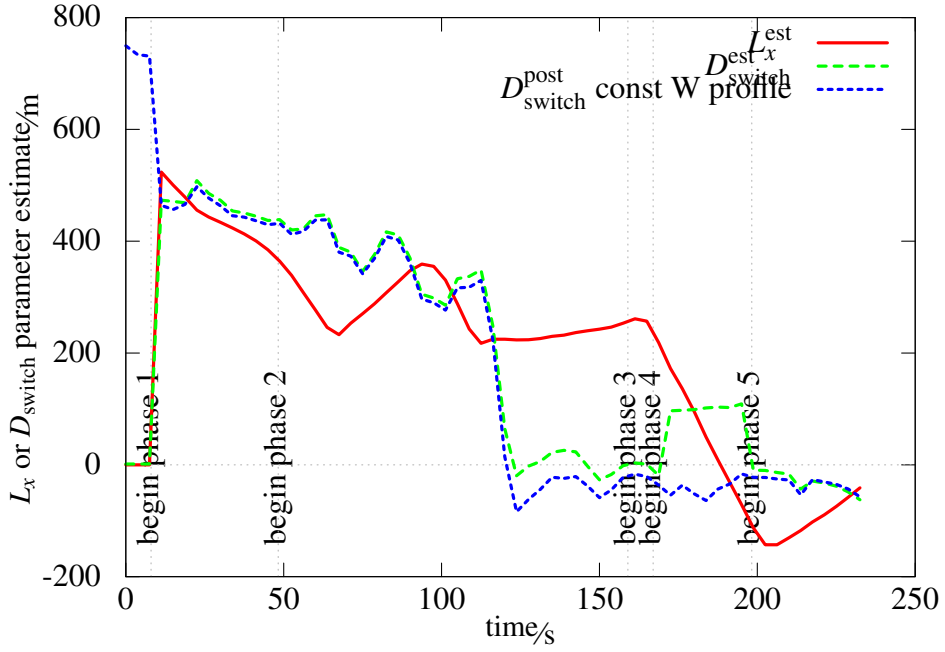


Figure 4.22: Final turn parameter estimates using logarithmic wind model. From a Snowflake test flight in May 2011, values of L_x and D_{switch} were estimated in flight and are plotted versus time. Values of D_{switch} assuming a constant wind profile were computed using a post-processing algorithm, and are also plotted versus time.

Figure 4.22 is the post-flight computation of D_{switch} using the constant wind profile. With the constant wind profile, the value of D_{switch} remained negative, indicating a turn prior to the abeam position.

One additional opportunity for comparison was available from these flight tests. The Snowflake drop 2, whose data were plotted in Figures 4.21 and 4.22, was actually dropped simultaneously with a UAH Snowflake that was programmed to use a constant wind profile. These two Snowflakes had the same target, and would fly through approximately the same wind field. Flight trajectories of both of these Snowflakes are compared in Figure 4.23 and Figure 4.24.

The Snowflake using the constant wind profile turned prior to the abeam point and ended up flying downwind of the target. The Snowflake using the logarithmic wind profile turned after the abeam point, and executed a typical final turn to the target, landing approximately 33 m upwind of the target. While the Snowflake that used the constant wind profile model

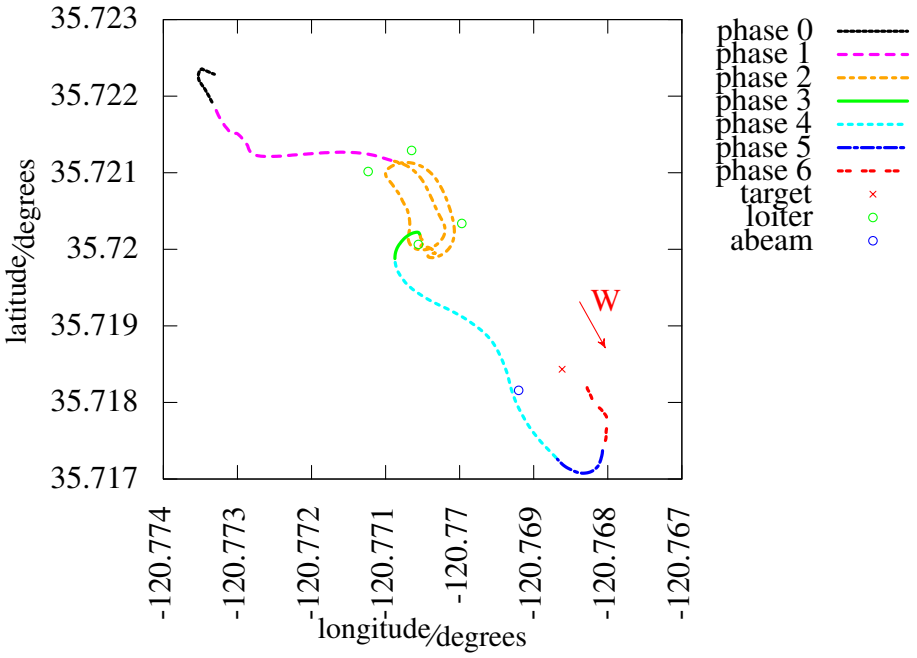


Figure 4.23: Parafoil trajectory overview using logarithmic wind profile. This trajectory is from a flight test conducted in May 2011.

did land closer to the target, its trajectory was less regular, since it had to enter an overhead spiraling approach to the target after flying over the target too high due to its early turn. For potential shipboard landing of an ADS, a predictable, non-spiraling approach trajectory is better.

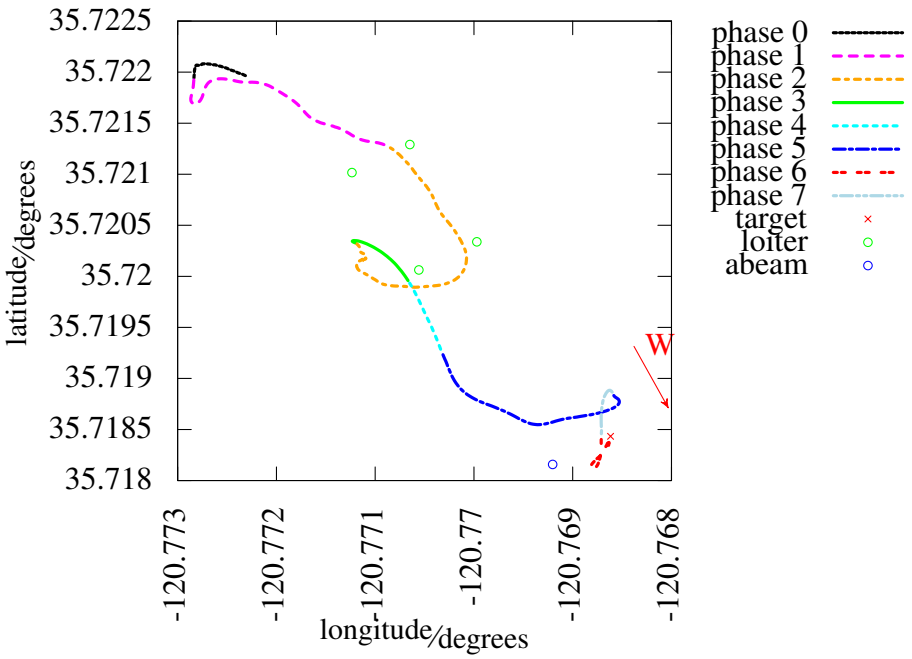


Figure 4.24: Side-by-side trajectory comparison using constant wind profile. This trajectory was flown in May 2011 by a Snowflake using a constant wind profile model that was released simultaneously with the Snowflake that flew the trajectory depicted in Figure 4.23.

THIS PAGE INTENTIONALLY LEFT BLANK

CHAPTER 5:
Conclusions and Future Work

The technical material in Chapters 3 and 4 serves as an indication of just how challenging is this problem of landing an ADS onto a moving target. Those chapters contained detailed discussions of applicable experimental results. Higher level conclusions derived from the preceding results are contained in this chapter.

5.1 Summary of Findings

The goal of this work was to identify and take the first steps on the development of technologies that will enable the landing of an autonomous ADS aboard a moving platform at sea. The primary methods of this research were simulation and direct experimentation; therefore, the findings follow from the experimental results detailed in Chapters 3 and 4. Specific experimental results include:

1. The visual estimation algorithm with epipolar constraint processing detailed in Chapter 3 was effective in simulation for estimating position and velocity of a target that was intermittently visible. The use of the epipolar constrained reduced Kalman estimator re-initialization errors when the target returned to view. The use of an UKF also led to faster convergence for the estimator.
2. The logarithmic wind model and its associated parameter estimation algorithm detailed in Chapter 4, when used in a 6DOF simulation, demonstrated an improvement in terminal guidance decisions and landing accuracy.

The simulations designed to evaluate the visual sensing algorithm demonstrated that estimation errors in the target's position, velocity, and heading all converged to small values in times of the order of 30 s. For operational use, a properly deployed ADS is likely to have at least 30 s during which the target is in view as the ADS approaches to land on a ship. Further experiments will indicate whether this assumption is borne out.

Simulations, including a 6DOF model of the Snowflake ADS, provided the analytical results for the wind modeling portion of this work. These experiments demonstrated that,

when the Snowflake guidance algorithm assumed a logarithmic wind profile, it made a more advantageous decision of when to initiate the final turn to the target than did a similar guidance algorithm that instead assumed a constant wind profile. Some flight testing of the logarithmic wind profile algorithm was conducted using a fixed target on land because a suitable experimental moving target was not yet available. These experiments reinforced the idea that the improvement in decision-making that the logarithmic wind profile affords should also benefit attempts to land on a moving platform at sea. Directly following this point, all research in this dissertation was conducted with the aim of enabling an ADS to land on a moving at-sea platform.

5.2 Contributions of this Work

The main contributions of this work was demonstration, through simulation and experimentation, of signal processing and sensing methods that should ultimately provide the basis for a moving target capability for autonomous parafoils. Specific contributions include:

1. Development of a target estimation algorithm featuring a novel, dual-rate estimation scheme that enables an ADS equipped with a monocular vision sensor to maintain track on a target that is intermittently out of view.
2. Development of a wind modeling algorithm that allows an ADS to incorporate its in-flight wind measurements into an RLS estimation scheme that computes the shape of a logarithmic wind profile which predicts horizontal wind velocities down to the surface.

These contributions represent a significant first step toward the investigation of enabling ADSs to land on moving platforms. To guide subsequent research into this topic, other directions of research that might prove fruitful for further development of the capability to land on a moving target are summarized in the following section.

5.3 Suggestions for Future Research

One exciting aspect of research involving aerial delivery systems is the immaturity of the field. Basic, meaningful improvements remain to be accomplished in many areas. The Snowflake ADS will continue to be a very valuable research tool with which to investigate

these new ideas. The following subsections contain brief summaries of some possible lines of research.

5.3.1 Conduct Additional Realistic Experimentation

Very preliminary moving target experiments conducted by Hewgley, Yakimenko, and Slegers [10] have added encouraging evidence that shipboard landing of an ADS is an achievable possibility. The coupling of small aerial delivery systems with unmanned aerial vehicles will extend the possible uses of ADSs in the maritime domain even further. In a recent description of research involving Snowflake and the Arcturus T-20 UAS, Yakimenko [39] proposed many new avenues of further research involving these systems. Certainly, much work remains to be done for modeling realistic shipboard landing platforms and characterizing the wind environments around these vessels as they are underway. To this end, an initial set of maritime experiments is being planned to be conducted at the U.S. Naval Academy (USNA). The scale flight deck on the instrumented YP vessel based at USNA described in research by Snyder [106] will provide an ideal maritime target for these experiments.

5.3.2 Improve Capabilities of Sensors for GN&C

The Snowflake ADS is such an important research tool due to the advanced sensors included in its autopilot; however, there are still many opportunities for improvement. Below are some ideas for continued development.

Synchronization of Visual Sensor Data and Other GN&C Sensor Data

The hybrid of flight test and simulation presented in Chapter 3 required that the recorded state information from Snowflake and the recorded video from the externally mounted camera be synchronized. This process was completed manually by using the audio recording from the camera and aligning sounds from the servo actuations with turn commands from the recorded state information. The synchronization process would have been easier and more accurate had the video and state information data streams both been referenced to a common time base such as GPS time. Johansen from BYU addressed a very similar problem in his work with miniature UASs [76]; his work was discussed briefly in Section 2.6.3. The next series of flight tests of the visual estimation algorithm on the Snowflake ADS

should certainly include some method of synchronizing the video data with the data streams from other on-board sensors.

Improved Sensing of Height Above Surface

Error in an ADS's estimate of its height above the landing target is a leading contributor to miss distance upon landing. This assertion as been borne out in flight tests of the Snowflake ADS conducted by the NPS and UAH team [11]. Uncertainty in landing deck height is likely to be especially troublesome in a maritime scenario when the landing target is a small, moving platform at some height above the surface. The work of Barber at BYU on visual height-above-touchdown estimation for miniature ADSs was briefly discussed in Section 2.6.2 and highlights the interesting possibility of incorporating visual height-above-touchdown sensing into Snowflake's guidance algorithm.

Additional Input to the ADS's Navigation Solution

The current version of the Snowflake ADS, like other current operational and prototype ADSs, relies primarily on GPS signals to determine its own position; however, Snowflake also uses barometric altimeter data to determine its height. For the scenario of shipboard landing, especially when the landing platform is assumed to have constant velocity, methods investigated at NPS for measuring the relative position between an airborne observer and a moving target on the surface may be applicable. These methods rely on using accelerometer data aboard the airborne observer; they were briefly discussed in Section 2.6.3. Future target estimation algorithm development for the Snowflake ADS should include an investigation into how accelerometer data could be incorporated into the visual sensing algorithm described in Chapter 3.

On-board Airspeed Sensing

The wind triangle was shown in Figure 4.1 in Section 4.1.1 wherein it was explained that the Snowflake ADS measures directly both the magnitude and direction of the ground vector and the direction only of the air vector. Direct measurement of the magnitude of the air vector (airspeed) would enable immediate computation of the third side of the wind triangle, the wind vector. Pitot-static sensors are commonly used in fixed-wing aircraft to measure airspeed; however, such sensors become unreliable at low airspeeds. Student research projects have begun at USNA to determine whether such sensors would be suitable for use aboard an ADS.

5.3.3 Enable External Sources of Wind Information

Even with the more in-depth experimentation and upgraded sensors proposed in the previous two sections, the fact remains that a small ADS like Snowflake is still at the mercy of the wind. The wind modeling algorithm described in Chapter 4 represents an improvement over previous methods, but it is no substitute for having actual data of wind conditions below the descending ADS provided by an external source. Previous NPS and UAH experiments using a ground-based wind sensor [113] should be extended. One possibility is *wind sharing*, in which, with more than one ADS in flight, the leading, lower ADS shares its wind information with the following, higher ADS, allowing that system to have better awareness of wind conditions between itself and the target.

5.3.4 Continue Development of Snowflake GN&C Algorithms

The final set of recommendations for future research are the most challenging and also the most important. The Snowflake prototype ADS was designed primarily as a platform for testing *algorithms*, and the main focus for future research should be in this area. Of the many possible avenues for algorithm research, a few that have potential to produce substantial improvements in landing performance are mentioned below.

Use Two-Dimensional Wind Estimation Algorithm

In Chapter 4, both processes of wind estimation and wind profile modeling were constrained to one dimension. The survey of wind estimation techniques presented in Section 4.1.1 described methods by Carter [38] and by Ward, Costello, and Slegers [103] that each solved the wind triangle for the wind vector in two dimensions. A two-dimensional wind estimation algorithm similar to one of these should be implemented in the Snowflake ADS.

Use Logarithmic Wind Profile for All Calculations

Section 4.4 and its subsequent subsections contained the derivation for the calculation of D_{switch} using the logarithmic wind profile. The next step should be to derive, using the logarithmic wind profile, the expressions for the loiter exit altitude z_{start} and the estimated final approach time T_{app}.

Use Processed Wind Estimates

There is another improvement that can be made in wind estimation. In Sections 4.6 and 4.7.1, it was explained that of the key guidance decisions, currently, the computation of the loiter exit altitude z_{start} is done using the constant wind profile assumption only. Currently, the wind value used for the constant profile assumption is that derived from the rudimentary 1D estimation algorithm described in Section 4.3.2 (actually, this value processed through a simple low-pass filter). The previous recommendation stated in Section 5.3.4 of using the logarithmic wind profile for this calculation is the best approach. As a prior step, it is possible to use the RLS-filtered wind magnitude value as the input value for the constant profile assumption.

Use IDVD in Target-Fixed Coordinates for Final Trajectory

The 6DOF simulation used in Chapter 4 was a modified version of one designed to evaluate the landing performance for a fixed target on land. Both of the available 6DOF simulations shared this characteristic. The next step in development should be to develop the optimal final turn trajectory algorithm using IDVD in a coordinate frame attached to the moving target. Section 1.1.1 contains a brief introduction to the IDVD technique. In this way, the optimal final turn trajectory can be planned in two dimensions to the landing area. Recent work of Yakimenko and Slegers [9] should be a guide in this effort. Instead of having a hard constraint for the final time of the trajectory, the ADS should reach its goal point at *or before* the instant that it descends to the landing platform height. Once the ADS has reached this point and the forward speed of the landing platform has been matched to that of the ADS, the ADS will simply glide straight down to achieve its shipboard landing.

With these suggestions for future research in mind, the conclusion of the work described in this dissertation is really a starting point for other directions of research into aerial delivery systems. It is my earnest hope that the target estimation and wind modeling methods described in this dissertation are meaningful first steps toward a moving target capability for autonomous parafoils.

APPENDIX A:
Code Listings

A.1 Van Loan Method for Computing Φ and Q

The following code listing is adapted from a code listing in Grewal and Andrews [90]:

```
M   = Ts*[-F,G*Qc*G';zeros(n),F'];  % Ts is fundamental sampling time
N   = expm(M);
Phi = N(n+1:2*n,n+1:2*n)';  % lower-right block of N
Q   = Phi*N(1:2,n+1:2*n);   % Phi times upper-right block of N
```

A.2 Computing D_{switch} Using Logarithmic Wind Profile

The following MATLAB script was used to calculate guidance parameter D_{switch} after the flight given a recorded series of in-flight parameters alpha and beta.

```
function D_switch = calc_Dswitch_log_W( altitude_AGL,vvelest,vssest,Lx,alpha,beta,R,D_switch0 )
% D_SWITCH = CALC_DSWITCH_LOG_W( ALTITUDE_AGL,VVELEST,VSSEST,LX,ALPHA,BETA,R,D_SWITCH0 )
% This function performs the iterative calculation of D_switch based on the
% assumption of a logarithmic wind profile.
%
% *** NOTE: this function is based on the *code* sign convention:
% vertical velocity is negative
% altitude          is positive
% wind              is positive ***
%
% Input parameters:
% altitude_AGL -- above-ground-level altitude [m]
% vvelest      -- estimated vertical velocity [m/s]
% vssest       -- estimated horizontal velocity [m/s]
% T_turn       -- estimated duration of final turn [s]
% Lx           -- distance in the 'x' direction from Snowflake to abeam
% target position [m]
% alpha,beta   -- log wind profile parameters calculated using RLS
% algorithm
% R            -- final turn radius [m]
%
% Global variables:
% D_SWITCH_ITER_TOL -- iteration tolerance for D_switch calculation
% D_SWITCH_MAX_ITER -- maximum number of iterations for D_switch
% calculation
global D_SWITCH_ITER_TOL D_SWITCH_MAX_ITER

%% initial conditions
```

```
T_turn = pi*R/vssest;  % time to complete 180 deg turn [s]
%% iteration setup
delta = D_switch0;  % difference value for comparison
D_switch(1) = D_switch0; % first value is the parameter which is calculated
% using the assumption of constant wind profile
i = 1;              % initialize loop counter
while (abs(delta) > D_SWITCH_ITER_TOL) && (i < D_SWITCH_MAX_ITER)
%% iteration step 1: select an initial guess value for D_switch
% this is done as an input parameter
%% iteration steps 2 and 3: compute z0, T_app using D_switch
% compute constants for product log equation
a = (vssest-alpha+beta)/vvelest;
b = alpha/vvelest;
c = Lx+D_switch(i)+(vssest-alpha+beta)/vvelest*altitude_AGL+(alpha/vvelest)*altitude_AGL*log(altitude_AGL);
z0 = c / ( b * lambertw( c * exp(a/b)/b ) );
T_app = -altitude_AGL/vvelest - T_turn - (z0 - altitude_AGL)/vvelest;
%% iteration steps 4 and 5: compute Dtilde using z0
% compute z1 using constant vertical velocity
z1 = z0 + vvelest*T_turn;
Dtilde = (alpha-beta)*T_turn + alpha/vvelest*( z0*log(z0) ...
- z1*log(z1) );
%% iteration step 6: compute L_app using T_app
% this calculation incorporates the assumption of estimated wind equals
% zero at altitudes less than the aerodynamic roughness length
L_app = (vssest + alpha - beta )*T_app + alpha*exp(-beta/alpha)/vvelest ...
- alpha*T_app*log(-T_app*vvelest);
%% iteration step 7: compute D_switch
%% ** NOTE: under the code sign convention, the fundamental equation is:
% (*) D_switch + Dtilde = L_app
D_switch(i+1) = L_app - Dtilde; %#ok<AGROW>
%% update iteration step
i = i + 1;
delta = D_switch(i) - D_switch(i-1);
end
% return only the last value of the iteration
D_switch = D_switch(end);
end
```

A.3 RLS Estimation of Wind Profile Parameters

The following MATLAB script was used to calculate the logarithmic wind profile values α and β using recorded Snowflake flight data. The logarithm (base ten) of the recorded altitude is used as the input sequence, and the sequence of in-flight horizontal wind speed measurements is used as the desired sequence.

```
function [y_hat,theta_hat] = myRLS(z,d,lambda,alpha0,beta0)
%% [Y_HAT,THETA_HAT] = MYRLS(Z,D,LAMBDA,ALPHA0,BETA0)
```

```
% Recursive Least Squares adaptive filter algorithm for wind estimation
% based on notation from [EC4440 Class Notes, Prof. M. Fargues]
%
% based on original code: crlesqua.m, programmed by: Dimitris Manolakis, 1999
% modified by: Chas Hewgley, Naval Postgraduate School
%
% VERSION 3 (2011-02-18): * change notation (only) to be in accordance with
%                           my notation, which (roughly) is from
%                           Prof. Cristi/[Gelb]
%                         * retain order of calculations; believe them
%                           to be more efficient.
% VERSION 2 (2011-02-08): * change calculations to be in accordance with
%                           notation from [EC4440 Class Notes,
%                           Prof. M. Fargues]
%
% Input parameters:
% z          = input sequence
% H'         = vector in measurement equation
% d          = desired sequence
% lambda     = forgetting factor
% y_hat      = filtered sequence
% theta_hat = estimated parameters of wind model
%             alpha0, beta0 are the initial values of these parameters
% xi         = a priori error
%
% NOTE: this function assumes input z and desired signal d are
% *real-valued* signals.

% local variables
M     = 2; % number of computed parameters fixed at two
N     = length(z);
delta = 1 - lambda; % this is a small positive constant with which to
                    % initialize the value of P

% initialization step
P = delta * eye(M); % first value of P matrix is small positive constant
                    % times identity matrix

% computation step
% complete the first iteration of the loop
% compute the initial value of measurement matrix H
H = [z(1) 1];
% form the initial parameter estimate vector
theta_hat0 = [alpha0; beta0];
% compute the 'adaptation gain vector' k in two steps:
% PART 1
% first compute the numerator, g
g = P*H';
% compute the adaptation gain vector K
```

```
K = g / (lambda+H*g);
% PART 2
% need to incorporate initial value of theta_hat0 = [alpha0; beta0]
% Therefore, calculate the first set of parameters
% theta_hat thus:
% compute the a priori error
xi(1) = d(1) - H*theta_hat0;
% PART 3
% compute the first value of the filter coefficients
% ** NOTE theta_hat stored as a row vector vice column vector **
theta_hat = theta_hat0' + K'*xi(1);
% PART 4
% update the inverse of deterministic correlation matrix P
P = P/lambda - (K*H*P)/lambda;
% the following computation ensures that the matrix P
% stays Hermitian
P=(P+P')/2;
% PART 5
% finally, compute the a posteriori value of the first filter output
% ** NOTE theta_hat stored as a row vector vice column vector **
y_hat = H * theta_hat';

% complete the remaining iterations of the loop
for n = 2:N
    % compute the measurement matrix H
    % *** NOTE *** measurement matrix H here doesn't contain present and past
    % values of z, but instead contains [z[n] 1] for this wind estimation
    % problem!
    % *** NOTE *** H is a row vector here
    H = [z(n) 1];
    % compute the 'adaptation gain vector' K in two steps:
    % PART 1
    % first compute the numerator, g
    g = P*H';
    % compute the adaptation gain vector K
    K = g / (lambda+H*g);
    % PART 2
    % compute the a priori error
    % ** NOTE theta_hat stored as a row vector vice column vector **
    xi(n) = d(n) - H*theta_hat(n-1,:)'; %#ok<AGROW>
    % PART 3
    % update filter coefficients
    % *** NOTE *** g' instead of g because theta_hat is represented as a
    % row vector
    theta_hat(n,:) = theta_hat(n-1,:) + K'*xi(n);
    % PART 4
    % update the inverse of deterministic correlation matrix P
    P = P/lambda - (K*H*P)/lambda;
    % the following computation ensures that the matrix P
```

```
        % stays Hermitian
        P=(P+P')/2;
103     % PART 5
        % finally, compute the a posteriori value of filter output
        y_hat(n) = H * theta_hat(n,:)';
106 end %for
  end %function
```

A.4 Processing Snowflake Recorded Data

This excerpt of the MATLAB script `read_snowflake_data.m` shows an example of how the integers from the tab-delimited text file stored in non-volatile memory by the Snowflake autopilot is re-scaled and converted to SI units.

```
453 %% Read and process raw data for other recorded parameters
  fix             = Data(range,16);
  hac             = (2^8*Data(range,17)+Data(range,18));
456 speed           = (2^8*Data(range,21)+Data(range,22));
  Autoflag        = Data(range,30);
  brake           = (2^8*Data(range,34)+Data(range,35));
459 Lxdata          = (2^8*Data(range,40)+Data(range,41));
  roll            = (2^8*Data(range,45)+Data(range,46));
  pitch           = (2^8*Data(range,47)+Data(range,48));
462 yaw             = (2^8*Data(range,49)+Data(range,50));
  ax              = (2^8*Data(range,51)+Data(range,52));
  ay              = (2^8*Data(range,53)+Data(range,54));
465 az              = (2^8*Data(range,55)+Data(range,56));
  p               = (2^8*Data(range,57)+Data(range,58));
  q               = (2^8*Data(range,59)+Data(range,60));
468 r               = (2^8*Data(range,61)+Data(range,62));
  aswitchdata     = (2^8*Data(range,65)+Data(range,66));
  dswitchdata     = (2^8*Data(range,67)+Data(range,68));
471 TargetID        = Data(range,71);
  target_baro   = (2^8*Data(range,84)+Data(range,85));
  % adjust range of data points using logical arrays;
474 % data values that have a greater positive value than the maximum value for
  % a twos complement number for that number of bits are invalid.  Convert
  % these values back to negative numbers.
477 brake(brake > MAX_TWOS_16_BIT)    = brake(brake > MAX_TWOS_16_BIT) - MAX_16_BIT;
  Lxdata(Lxdata > MAX_TWOS_16_BIT) = Lxdata(Lxdata > MAX_TWOS_16_BIT) - MAX_16_BIT;
  roll(roll > MAX_TWOS_16_BIT)     = roll(roll > MAX_TWOS_16_BIT) - MAX_16_BIT;
480 pitch(pitch > MAX_TWOS_16_BIT)   = pitch(pitch > MAX_TWOS_16_BIT) - MAX_16_BIT;
  yaw(yaw > MAX_TWOS_16_BIT)       = yaw(yaw > MAX_TWOS_16_BIT) - MAX_16_BIT;
  ax(ax > MAX_TWOS_16_BIT)         = ax(ax > MAX_TWOS_16_BIT) - MAX_16_BIT;
483 ay(ay > MAX_TWOS_16_BIT)         = ay(ay > MAX_TWOS_16_BIT) - MAX_16_BIT;
  az(az > MAX_TWOS_16_BIT)         = az(az > MAX_TWOS_16_BIT) - MAX_16_BIT;
  p(p > MAX_TWOS_16_BIT)           = p(p > MAX_TWOS_16_BIT) - MAX_16_BIT;
```

```
q(q > MAX_TWOS_16_BIT)             = q(q > MAX_TWOS_16_BIT) - MAX_16_BIT;
r(r > MAX_TWOS_16_BIT)             = r(r > MAX_TWOS_16_BIT) - MAX_16_BIT;
aswitchdata(aswitchdata > MAX_TWOS_16_BIT) = aswitchdata(aswitchdata > MAX_TWOS_16_BIT) - MAX_16_BIT;
dswitchdata(dswitchdata > MAX_TWOS_16_BIT) = dswitchdata(dswitchdata > MAX_TWOS_16_BIT) - MAX_16_BIT;
target_baro(target_baro > MAX_TWOS_16_BIT) = target_baro(target_baro > MAX_TWOS_16_BIT) - MAX_16_BIT;
%% Apply scale factors and SI unit conversion to other recorded parameters
hac          = CONVERT_FT_M*hac/10;       % [m]
speed        = CONVERT_FT_M*speed/10;     % [m/s]
Lxdata       = CONVERT_FT_M*Lxdata;       % [m]
roll         = roll*pi/MAX_TWOS_16_BIT;   % [rad]   maximum value = pi
pitch        = pitch*pi/MAX_TWOS_16_BIT;  % [rad]   maximum value = pi
yaw          = yaw*pi/MAX_TWOS_16_BIT;    % [rad]   maximum value = pi
ax           = ax*8/MAX_TWOS_16_BIT;      % [g]     maximum value = 8
ay           = ay*8/MAX_TWOS_16_BIT;      % [g]     maximum value = 8
az           = az*8/MAX_TWOS_16_BIT;      % [g]     maximum value = 8
p            = p*10/MAX_TWOS_16_BIT;      % [rad/s] maximum value = 10
q            = q*10/MAX_TWOS_16_BIT;      % [rad/s] maximum value = 10
r            = r*10/MAX_TWOS_16_BIT;      % [rad/s] maximum value = 10
aswitchdata  = CONVERT_FT_M*aswitchdata;  % [m]
dswitchdata  = CONVERT_FT_M*dswitchdata;  % [m]
target_baro  = CONVERT_FT_M*target_baro;  % [m]
```

A.5 Archiving Snowflake Data as a MATLAB Structure

This excerpt of the MATLAB script `read_snowflake_data.m` shows an example of how the the post-processed Snowflake data is stored as a structure variable in MATLAB.

```
%% Transfer variables to data structure
sfdata.notes                  = notes;
sfdata.data.time              = time;              % [s]
sfdata.data.baro_altitude     = baro;              % [m]
sfdata.data.latitude          = lat;               % [deg]
sfdata.data.longitude         = lon;               % [deg]
sfdata.data.h_velocity_ss     = vssest;            % [m/s]
sfdata.data.v_velocity_ss     = vvelest;           % [m/s]
sfdata.data.horiz_magnitude   = windest;           % [m/s]
sfdata.data.vert_magnitude    = vvelest - vertvel; % [m/s]
sfdata.data.fix               = fix;
sfdata.data.HAC               = hac;               % [m]
sfdata.data.GPS_velocity      = speed;             % [m/s]
sfdata.data.autoflag          = Autoflag;
if vvelest_ovrw
    sfdata.data.integralerror = integralerror;
end % if vvelest_ovrw
sfdata.data.brake             = brake;
sfdata.data.Lx                = Lxdata;            % [m]
sfdata.data.roll              = roll;              % [rad]
sfdata.data.pitch             = pitch;             % [rad]
```

```
    sfdata.data.yaw                = yaw;                % [rad]
    sfdata.data.ax                 = ax;                 % [g]
492 sfdata.data.ay                 = ay;                 % [g]
    sfdata.data.az                 = az;                 % [g]
    sfdata.data.p                  = p;                  % [rad/s]
495 sfdata.data.q                  = q;                  % [rad/s]
    sfdata.data.r                  = r;                  % [rad/s]
    sfdata.data.aswitch            = aswitchdata;        % [m]
498 sfdata.data.dswitch            = dswitchdata;        % [m]
    sfdata.data.targetID_cmd       = TargetID;
    if tgtwind_ovrw
501     sfdata.data.RLS_alpha      = RLS_alpha;
        sfdata.data.RLS_beta       = RLS_beta;
    else
504     sfdata.data.targetwindspeed = targetwindspeed; % [m/s]
        sfdata.data.targetwinddir = targetwinddir;     % [deg true]
    end % if tgtwind_ovrw
507 sfdata.data.target_baro         = target_baro;        % [m]
    sfdata.GNC.phase1              = find(pathindex==1,1,'first');
    sfdata.GNC.phase2              = find(pathindex==2,1,'first');
510 sfdata.GNC.phase3              = find(pathindex==3,1,'first');
    sfdata.GNC.phase4              = find(pathindex==4,1,'first');
    sfdata.GNC.phase5              = find(pathindex==5,1,'first');
513 % Phase 6 and phase 7 can be entered multiple times.  The following code
    % parses the transitions between phase 6 and phase 7, and stores the result
    % as vectors in the phase6 and phase7 variables.  These variables will hold
516 % indices of the transitions *into* those phases.
    %
    % form a simple difference filter: y[n] = x[n+1] - x[n]
519 path_filt = filter([1 -1],1,pathindex);
    % find the transition points where the filter output is non-zero
    transitions = find(path_filt,NUM_TRANSITIONS,'last');
522 % remove the first few transitions found if they are at or before the
    % transition to phase 5
    n = 1;
525 while transitions(n) < sfdata.GNC.phase5
        transitions = transitions(n+1:end);
        n = n + 1;
528 end % while
    % use logical arrays to put the appropriate transitions into the proper
    % variables
531 sfdata.GNC.phase6               = transitions((pathindex(transitions) == 6));
    sfdata.GNC.phase7              = transitions((pathindex(transitions) == 7));
```

THIS PAGE INTENTIONALLY LEFT BLANK

APPENDIX B:
System Development and Flight Test

Chapters 3 and 4 contain theoretical development and technical findings for the problems of target motion estimation and wind profile estimation. This chapter contains details of the flight testing that was performed between 2009 and 2011. The collection of flight test data to support the investigations described in Chapters 3 and 4 was a vital element of this research.

B.1 Snowflake System History

This section contains a brief summary of the origin and subsequent evolution of the Snowflake prototype ADS. Later, as different development teams designed Snowflake versions to be carried aloft by different vehicles, they created different sizes and shapes of Snowflake. Issues arising from these changes will be discussed at the end of the section.

B.1.1 System Origination

Professor Nathan J. Slegers (UAH) and Professor Oleg A. Yakimenko (NPS) designed the Snowflake ADS to be a small, easily deployable testbed for the development of ADS GN&C *algorithms*, not necessarily specific data processing hardware or sensors. When Slegers and Yakimenko began Snowflake development in late 2007, U.S. Army's NSRDEC was pursuing research into implementing simple control algorithms into a reliable system that would be compatible with current military airdrop infrastructure. Draper Laboratory developed a PID control algorithm for the system—see D. Carter *et al.* [36]–[38]. This control algorithm was included in JPADS [1], [34], which comprises a guidance unit, flight hardware, and a mission planning system. The result of this effort was a simple algorithm controlling a complex system, and overall high reliability. With Snowflake, the NPS–UAH team developed a complex algorithm for a simple system, and therefore was able to concentrate on developing the algorithm rather than on designing robust hardware.

B.1.2 System Evolution

The NPS–UAH team has developed Snowflake in an iterative fashion, with frequent flight tests shaping subsequent stages of development. After the initial sets of flight tests in

2008, Bourakov developed a ground-based networking and wind information system [113], which was tested in 2009. In 2010, Yingling and Seigenthaler developed a two-stage rocket that housed a cylindrical Snowflake ADS in the upper stage [114]. The work herein on wind estimation and visual sensing used flight test data from 2011 [10], [18]. Also in 2011, Benton at NASA Ames Research Center developed a Snowflake system designed to be released from a high-altitude balloon [115]. These varied designs are all members of the Snowflake family: simple systems that explore the frontier of possibility for aerial delivery.

B.1.3 Mass Properties

Even though the shapes and sizes of Snowflake systems vary, the guidance algorithm assumes a certain set of aerodynamic coefficients and mass properties for the Snowflake vehicle. These constants appear in the article by Slegers and Yakimenko [7]; refer to previous work by Slegers and Costello [8] for the method of parameter estimation used. Subsequent Snowflake designs have maintained the same model parafoil canopy (from the same vendor) in order to limit differences in aerodynamic and mass properties caused by the differences in size and mass of the various Snowflake bodies.

B.2 Snowflake Data Recording

B.2.1 On-board Sensors

Autonomous parafoils typically have an on-board processor that uses sensor data to generate commands to the control actuators. For JPADS and other large systems designed to deliver cargo pallets, the processor and sensors are housed in an AGU, which is an enclosure suspended between the cargo and the parafoil canopy. Smaller systems, such as Snowflake and the *Mosquito* system made by STARA Technologies, Inc. have an autopilot incorporated within the vehicle body.

For early versions of JPADS, the only sensor housed in the AGU was GPS (see Carter [36]). In contrast, the Snowflake prototype ADS was designed around a small autopilot unit that contained various sensors, including a three-axis magnetometer, a three-axis accelerometer, a three-axis angular rate sensor, and a barometric altimeter, in addition to having a GPS receiver (see Ref. 11 for details). Snowflake's autopilot was designed and built by Dr. Nathan Slegers of UAH, and is informally known as the Advanced Autopilot System (AAS). Other versions of Snowflake, such as those designed for the rocket experiments of Yingling [114],

or the high-altitude balloon deployments of Benton [115] have used the Monkey Autopilot from Ryan Mechatronics.

B.2.2 Data Post-processing

Data from the sensors on the AAS are recorded to non-volatile memory and then written to a text file after the flight. The text file contains tab-delimited integer numbers on each line, and each line is a separate record. Post-processing scripts written in MATLAB are used to convert the integers to floating-point numbers; and, if necessary, to perform a unit conversion from US Customary to SI units. An excerpt from this post-processing script is shown in Section A.4.

B.2.3 Structures for Archived Data

For further analysis, the post-processed Snowflake data was saved as a MATLAB structure variable. Subsequent analysis and plotting scripts would use the structure variable as an input. An example of some of the fields in this data structure is shown in the code listing in Section A.5.

B.3 Flight Test Chronology

Flight test has been the primary mode of system analysis for the Snowflake prototype ADS and also the basis for design changes since the project's inception in 2008. Table B.1 contains a chronological list of flight test events from 2008 to 2011. The three flight test topics that will be discussed herein are: system development, the logarithmic wind profile experiments, and the preliminary moving target experiments.

B.3.1 System Development

The flight test events in 2008 focused on verifying the procedures for preparing the system for flight and for deploying it from an aircraft. The NPS and UAH research team conducted flight tests to measure the landing accuracy to a fixed target on land. These tests were also useful for determining whether two key algorithms were implemented properly in the autopilot software code. The two algorithms were the MPC algorithm for following a planned flight path, and the IDVD algorithm for planning an optimal final turn trajectory to the target.

B.3.2 Wind Profile Experiments

Development of the logarithmic wind profile algorithm began in October 2010. Subsequently, the first version of the algorithm was flight tested at McMillan Airfield, Camp Roberts, California, in February 2011. At this stage of development, the autopilot program used the RLS estimation algorithm in flight to estimate the logarithmic wind profile parameters α and β; however, the program still used the constant wind profile assumption, and not the logarithmic wind profile assumption, to calculate the important guidance parameter D_{switch}. Therefore, from the February series of tests, final turn initiation altitudes associated with the logarithmic wind profile were computed on the ground, after the flight, using a MATLAB script (see listing in Section A.2).

For the subsequent set of flight tests in May 2011, the autopilot software was capable of both using RLS estimation to calculate the logarithmic wind profile parameters, and using these parameters to calculate D_{switch} in flight. This development made possible the direct comparison of the use of the constant wind profile versus the use of the logarithmic wind profile by dropping two Snowflakes simultaneously, each programmed to use a different wind profile assumption. In order to achieve a good comparison between the logarithmic wind profile assumption and the constant wind profile assumption, on each flight of the Arcturus T-20 UAS, the two Snowflakes were dropped, one from underneath each wing, as quickly as the two under-wing release mechanisms could be opened. Usually, both Snowflakes were dropped within one second of each other.

B.3.3 Moving Target Experiments

Another objective of the flight tests both in February and in May was to start preliminary work toward the objective of landing on a moving target. In February 2011, the development of the visual estimation algorithm described in Chapter 3 had not yet begun; therefore, the test team decided to start with a very simple scenario in which the moving target traveled slowly in a straight line along the runway at McMillan airfield. For both sets of flight tests, the moving target was equipped with a GPS beacon that transmitted the vehicle's position to the Snowflake once every two seconds. For the February tests, the moving target was a four-wheeled robot approximately 1 m in length. For the May tests, one member of the test team drove an automobile on the runway as a target.

One weakness of these tests was that there was no source of external truth data for the positions of the target vehicles other than GPS. The track of the target vehicle could have been fixed quite accurately using reference marks along the runway, and recording the time at which the target vehicle passed each mark.

To prepare for the development of the visual estimation algorithm, the test team decided for the May flight tests to attach a small camera externally to the Snowflake case. This camera was not connected in any way with the autopilot; its sole purpose was capturing video in flight for later analysis. As described in Chapter 3, a pinhole camera model can be used to form a relationship between the coordinates (x, y, z) of an object in space and the coordinates (u, v) of the object's image on the camera's image plane. One aspect of this relationship not often explicitly stated in published literature is that the image plane coordinates (u, v) are in units of length, *not pixels*. Therefore, the size of the imaging sensor itself must be known. For the GoPro camera used in the May 2011 tests, an estimate of the size of the image plane was made based on a typical size for the charge-coupled device (CCD) image sensor found in small, hand-held digital cameras.

Using the recorded video for useful analysis proved to be much more of a challenge. The visual estimation algorithm described in Chapter 3 requires information about the camera's position and orientation at the time of the image. For the May 2011 flight tests, there was no time information associated with the video stream; thus, there was no easy way to synchronize the video with the recorded data stream from Snowflake's other sensors. One method I tried to synchronize the two data streams involved using the audio track that was recorded by the camera and matching the sounds of the servomotor movements with the servomotor control signals recorded with the Snowflake data. This method was difficult, time-consuming, and inaccurate; as a result, the shortcomings of this method were the impetus for the recommendation made in Chapter 5 for better sensor design.

Table B.1: Chronological history of Snowflake flight tests. The Snowflake prototype ADS has been the centerpiece of a collaborative NPS and UAH research effort spanning several years.

testing location	dates	no. of drops	wind truth	research objective	tech. report	publication
	2008					
McMillan Airfield (CA62)	15 May	4	radiosonde	system development	TNT 08-3	Yakimenko, Slegers, Tiaden [11]
Sidewinder DZ, YPG	20–21 Oct.	16	wind tower windpack	system development		Yakimenko, Slegers, Tiaden [11]
	2009					
McMillan Airfield (CA62)	24 Feb.	4		GSM system development	TNT 09-2	Bourakov, Yakimenko, Slegers [113]
Sidewinder DZ, YPG	18–19 May	11	windpack	algorithm/GSM sys. devel.	TNT 09-3	
Red Lake DZ, Kingman, AZ	20–21 May	3	dropsonde	high-alt. drop from C-123	TNT 09-3	
McMillan Airfield (CA62)	3–6, 12 Aug.	12		pod configuration UAS integration	TNT 09-4	Yakimenko *et al.* [39]
Marina Muni. (OAR)	10 Aug.	2		carbon-fiber case test	TNT 09-4	
McMillan Airfield (CA62)	7 Oct.	5		PATCAD preparation	TNT 10-1	
Robby DZ, YPG	20–22 Oct.	6		incl. 1 Rascal UAS drop	PATCAD 09	Yakimenko *et al.* [39]
	2010					
McMillan Airfield (CA62)	4–6 May	17		GSM system testing larger canopy, robot	TNT 10-3	
Koehn Lake Test Area, CA	17 Jul.	1		rocket deployment		Yingling *et al.* [114]
McMillan Airfield (CA62)	9–10 Aug.	8		battlefield medic demo.	TNT 10-4	
Del Norte Launch Site, CA	16 Oct.	1		rocket deployment		Yingling *et al.* [114]
	2011					
McMillan Airfield (CA62)	21–24 Feb.	28		log wind profile canopy instrumentation		Hewgley, Yakimenko [18]
McMillan Airfield (CA62)	2–4 May	16		log wind profile moving target		Hewgley, Yakimenko, Slegers [10]

LIST OF REFERENCES

[1] R. Benney, M. Henry, K. Lafond, and A. Meloni, "DoD new JPADS programs and NATO activities," in *Proceedings of the 20th Aerodynamic Decelerator Systems Technology Conference*. Seattle, WA: American Institute of Aeronautics and Astronautics, 4–7 May 2009, paper AIAA 2009-2952.

[2] K. Lafond, R. Benney, M. Henry, A. Meloni, C. Ormonde, G. Noetscher, S. Patel, M. Shurtliff, S. Tavan, A. Goldenstein, and G. Pinnell, "Joint medical distance support and evacuation joint capability technology demonstration," in *21st AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar*, American Institute of Aeronautics and Astronautics. Dublin, Ireland: American Institute of Aeronautics and Astronautics, 23–26 May 2011. [Online]. Available: http://dx.doi.org/10.2514/6.2011-2574

[3] O. A. Yakimenko, "Direct method for rapid prototyping of near-optimal aircraft trajectories," *Journal of Guidance, Control, and Dynamics*, vol. 23, no. 5, pp. 865–875, Sep.–Oct. 2000. [Online]. Available: http://dx.doi.org/10.2514/2.4616

[4] I. I. Kaminer and O. A. Yakimenko, "Development of control algorithm for the autonomous gliding delivery system," in *Proceedings of the 17th Aerodynamic Decelerator Systems Technology Conference*. Monterey, CA: American Institute of Aeronautics and Astronautics, 19–22 May 2003, paper AIAA 2003-2116.

[5] I. Kaminer, O. Yakimenko, and A. Pascoal, "Coordinated payload delivery using high glide parafoil systems," in *18th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar*. Munich, Germany: American Institute of Aeronautics and Astronautics, 23–26 May 2005. [Online]. Available: http://dx.doi.org/10.2514/6.2005-1622

[6] N. J. Slegers and O. A. Yakimenko, "Optimal control for terminal guidance of autonomous parafoils," presented at the Proceedings of the 20th Aerodynamic Decelerator Systems Technology Conference. Seattle, WA: American Institute of Aeronautics and Astronautics, 4–7 May 2009, paper AIAA-2009-2958.

[7] N. Slegers and O. A. Yakimenko, "Terminal guidance of autonomous parafoils in high wind-to-airspeed ratios," *Proceedings of the Institution of Mechanical Engineers, Part G: Journal of Aerospace Engineering*, vol. 225, no. 3, pp. 336–346, Mar. 2011. [Online]. Available: http://pig.sagepub.com/content/225/3/336.abstract

[8] N. Slegers and M. Costello, "Model predictive control of a parafoil and payload system," *Journal of Guidance, Control, and Dynamics*, vol. 28, no. 4, pp. 816–821, Jul.–Aug. 2005. [Online]. Available: http://dx.doi.org/10.2514/1.12251

[9] O. A. Yakimenko and N. J. Slegers, "Optimization of the ads final turn maneuver in 2d and 3d," in *Proceedings of the 21st Aerodynamic Decelerator Systems Technology Conference*. Dublin, Ireland: American Institute of Aeronautics and Astronautics, 23–26 May 2011.

[10] C. W. Hewgley, O. A. Yakimenko, and N. J. Slegers, "Shipboard landing challenges for autonomous parafoils," in *Proceedings of the 21st Aerodynamic Decelerator Systems Technology Conference*. Dublin, Ireland: American Institute of Aeronautics and Astronautics, 23–26 May 2011, paper AIAA 2011-2573, pp. 846–855.

[11] O. A. Yakimenko, N. J. Slegers, and R. Tiaden, "Development and testing of the miniature aerial delivery system snowflake," in *Proceedings of the 20th Aerodynamic Decelerator Systems Technology Conference*. Seattle, WA: American Institute of Aeronautics and Astronautics, 4–7 May 2009, paper AIAA-2009-2980.

[12] M. L. Salby, *Fundamentals of Atmospheric Physics*, ser. International Geophysics Series. San Diego, CA: Academic Press, 1996, vol. 61.

[13] R. B. Stull, *Meteorology for Scientists and Engineers*, 2nd ed. Pacific Grove, CA: Brooks/Cole, 2000.

[14] U.S. Navy, "APP 2(F)/MPP 2(F) Volume I helicopter operations from ships other than aircraft carriers (HOSTAC)," Allied Publication / Multinational Manual, Sep. 2001.

[15] U.S. Army, "Precision airdrop technology conference and demonstration 2009, final report," U.S. Army Research, Development, and Engineering Command,

Natick Soldier Research, Development and Engineering Center (NSRDEC), Tech. Rep., Apr. 2010.

[16] U.S. Department of Defense, "Dictionary of military and associated terms," Joint Publication 1-02, Apr. 2001, amended through 31 October 2009.

[17] C. W. Hewgley and O. A. Yakimenko, "Precision guided airdrop for vertical replenishment of naval vessels," in *Proceedings of the 20th Aerodynamic Decelerator Systems Technology Conference*. Seattle, WA: American Institute of Aeronautics and Astronautics, 4–7 May 2009, paper AIAA-2009-2995.

[18] C. W. Hewgley and O. A. Yakimenko, "Improved surface layer wind modeling for autonomous parafoils in a maritime environment," in *Proceedings of the 21st Aerodynamic Decelerator Systems Technology Conference*. Dublin, Ireland: American Institute of Aeronautics and Astronautics, 23–26 May 2011, paper AIAA 2011-2605.

[19] S. E. Anders, "Bundles from the sky," *Quartermaster Professional Bulletin*, pp. 42–45, Autumn-Winter 1994.

[20] R. E. Bilstein, *Airlift and Airborne Operations in World War II*. Washington, DC: Air Force History and Museums Program, 1998.

[21] J. A. Huston, *Out of the Blue; U.S. Army Airborne Operations in World War II*. West Lafayette, Ind.: Purdue University Studies, 1972.

[22] H. M. Cole, *The Ardennes: Battle of the Bulge*, ser. The United States Army in World War II. Washington, D.C: Center of Military History, U.S. Army, 1993, vol. 7-8. [Online]. Available: http://www.history.army.mil/html/books/007/7-8-1/index.html

[23] U.S. Army Quartermaster Foundation, Inc. Quartermaster aerial delivery; the story of the airborne rigger. [Online]. Available: http://www.qmfound.com/riggers.htm

[24] T. G. Kistler. Airmen airdrop relief supplies to Haitians. [Online]. Available: http://www.af.mil/News/ArticleDisplay/tabid/223/Article/117772/airmen-airdrop-relief-supplies-to-haitians.aspx

[25] S. Dellicker, R. Benney, S. Patel, T. Williams, C. Hewgley, O. Yakimenko, R. Howard, and I. Kaminer, "Performance, control, and simulation of the Affordable Guided Airdrop System," presented at the AIAA Modeling and Simulation Technologies Conference. Denver, CO: American Institute of Aeronautics and Astronautics, Aug. 2000, paper AIAA-2000-4309.

[26] O. Yakimenko, V. Dobrokhodov, I. Kaminer, and S. Dellicker, "Synthesis of optimal control and flight testing of an autonomous circular parachute," *Journal of Guidance, Control, and Dynamics*, vol. 27, no. 1, pp. 29–40, Jan.–Feb. 2004. [Online]. Available: http://dx.doi.org/10.2514/1.9282

[27] S. Tavan, "Status and context of high altitude precision aerial delivery systems," in *Proceedings of the Guidance, Navigation, and Control Conference and Exhibit*. Keystone, CO: American Institute of Aeronautics and Astronautics, 21–24 Aug. 2006, paper AIAA-2006-6793.

[28] British Broadcasting Corporation. Who do pirates call to get their cash? [Online]. Available: http://news.bbc.co.uk/2/hi/uk_news/magazine/7847351.stm

[29] E. D. Harrison. Navy ship recovers special operations gear during exercise. [Online]. Available: http://www.eucom.mil/media-library/article/20880/navy-ship-recovers-special-operations-gear

[30] T. Cross and S. C. Truver, "Endangered species," *Proceedings*, vol. 137, no. 8, pp. 28–33, Aug. 2011.

[31] U.S. Navy, "Naval S&T strategic plan: Defining the strategic direction for tomorrow," Office of Naval Research, Washington, DC, Tech. Rep., Jan. 2007.

[32] R. Benney, A. Meloni, M. Henry, K. Lafond, G. Cook, S. Patel, and L. Goodell, "Joint medical distance support and evaluation (JMDSE) joint capability technology demonstration (JCTD) & joint precision air delivery systems (JPADS)," presented at the Special Operations Forces Industry Conference, Tampa, FL, 2–4 Jun. 2009, paper 8304.

[33] R. Benney, J. Barber, J. McGrath, J. McHugh, G. Noetscher, and S. Tavan, "The joint precision airdrop system advanced concept technology demonstration," in

18th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar, American Institute of Aeronautics and Astronautics. Munich, Germany: American Institute of Aeronautics and Astronautics, 23–26 May 2005. [Online]. Available: http://dx.doi.org/10.2514/6.2005-1601

[34] R. Benney, J. McGrath, J. McHugh, G. Noetscher, S. Tavan, and S. Patel, "DOD JPADS programs overview and NATO activities," in *Proceedings of the 19th Aerodynamic Decelerator Systems Technology Conference and Seminar*. Williamsburg, VA: American Institute of Aeronautics and Astronautics, 21–24 May 2007, paper AIAA 2007-2576.

[35] S. Tavan, A. Dietz, P. Sorenson, G. Noetsche, C. McCavitt, and G. Bailey, "Advanced sensors for precision airdrop," presented at the 20th AIAA Aerodynamic Decelerator Systems Technology Conference. Seattle, WA: American Institute of Aeronautics and Astronautics, 4–7 May 2009, paper AIAA-2009-2951.

[36] D. Carter, S. George, P. Hattis, L. Singh, and S. Tavan, "Autonomous guidance, navigation and control of large parafoils," in *Proceedings of the 18th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar*. Munich, Germany: American Institute of Aeronautics and Astronautics, 23–26 May 2005, paper AIAA-2005-1643. [Online]. Available: http://dx.doi.org/10.2514/6.2005-1643

[37] D. W. Carter, S. George, P. D. Hattis, M. W. McConley, S. A. Rasmussen, L. Singh, and S. Tavan, "Autonomous large parafoil guidance, navigation, and control system design status," in *Proceedings of the 19th Aerodynamic Decelerator Systems Technology Conference and Seminar*. Williamsburg, VA: American Institute of Aeronautics and Astronautics, 21–24 May 2007, paper AIAA-2007-2514. [Online]. Available: http://dx.doi.org/10.2514/6.2007-2514

[38] D. Carter, L. Singh, L. Wholey, M. McConley, S. Tavan, B. Bagdonovich, T. Barrows, C. Gibson, S. George, and S. Rasmussen, "Band-limited guidance and control of large parafoils," in *20th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar*, American Institute of Aeronautics and Astronautics. Seattle, WA: American Institute of Aeronautics and Astronautics, 4–7

May 2009, paper AIAA-2009-2981. [Online]. Available: http://dx.doi.org/10.2514/6.2009-2981

[39] O. A. Yakimenko, E. A. Bourakov, C. W. Hewgley, N. J. Slegers, R. P. Jensen, A. B. Robinson, J. R. Malone, and P. E. Heidt, "Autonomous aerial payload delivery system Blizzard," in *Proceedings of the 21st Aerodynamic Decelerator Systems Technology Conference*. Dublin, Ireland: American Institute of Aeronautics and Astronautics, 23–26 May 2011, paper AIAA 2011-2594. [Online]. Available: http://dx.doi.org/10.2514/6.2011-2594

[40] B. J. Rademacher, P. Lu, A. L. Strahan, and C. J. Cerimele, "In-flight trajectory planning and guidance for autonomous parafoils," *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 6, pp. 1697–1712, Nov.–Dec. 2009.

[41] L. E. Fowler and J. D. Rogers, "Bézier curve path planning for parafoil terminal guidance," in *Proceedings of the 22nd AIAA Aerodynamic Decelerator Systems (ADS) Conference*. Daytona Beach, FL: American Institute of Aeronautics and Astronautics, 25–28 Mar. 2013. [Online]. Available: http://dx.doi.org/10.2514/6.2013-1325

[42] J. D. Rogers and N. Slegers, "Terminal guidance for complex drop zones using massively parallel processing," in *Proceedings of the 22nd AIAA Aerodynamic Decelerator Systems (ADS) Conference*. Daytona Beach, FL: American Institute of Aeronautics and Astronautics, 25–28 Mar. 2013. [Online]. Available: http://dx.doi.org/10.2514/6.2013-1343

[43] K. Kelly and B. Peña, "Wind study and GPS dropsonde applicability to airdrop testing," in *Proceedings of the 16th Aerodynamic Decelerator Systems Technology Conference and Seminar*. Boston, MA: American Institute of Aeronautics and Astronautics, 21–24 May 2001. [Online]. Available: http://dx.doi.org/10.2514/6.2001-2022

[44] J. Rogers, "Comparative analysis involving wind profile data sources," in *Proceedings of the 20th Aerodynamic Decelerator Systems Technology Conference and Seminar*. American Institute of Aeronautics and Astronautics, 4–7 May 2009. [Online]. Available: http://dx.doi.org/10.2514/6.2009-2960

[45] R. Fraser, “Analysis of windpack meteorogical data,” in *Proceedings of the 21st AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar*. Dublin, Ireland: American Institute of Aeronautics and Astronautics, 23–26 May 2011. [Online]. Available: http://dx.doi.org/10.2514/6.2011-2515

[46] T. A. Herrmann, M. B. Ward, M. Costello, and N. Slegers, “Utilizing ground-based LIDAR for autonomous airdrop,” in *Proceedings of the 22nd Aerodynamic Decelerator Systems (ADS) Conference*. American Institute of Aeronautics and Astronautics, 25–28 Mar. 2013, paper AIAA-2013-1387. [Online]. Available: http://dx.doi.org/10.2514/6.2013-1387

[47] K. Bergeron, A. Fejzic, and S. Tavan, “Accuglide 100: Precision airdrop guidance and control via glide slope control,” in *Proceedings of the 21st Aerodynamic Decelerator Systems Technology Conference*. Dublin, Ireland: American Institute of Aeronautics and Astronautics, 23–26 May 2011, paper AIAA 2011-2530.

[48] K. Bergeron, M. Ward, M. Costello, and S. Tavan, “Accuglide 100 and bleed-air actuator airdrop testing,” in *Proceedings of the 22nd AIAA Aerodynamic Decelerator Systems (ADS) Conference*. American Institute of Aeronautics and Astronautics, 25–28 Mar. 2013, paper AIAA-2013-1378. [Online]. Available: http://dx.doi.org/10.2514/6.2013-1378

[49] H. Altmann, “An enhanced GNC functionality combining pre-flight wind forecast and in-flight identified wind,” in *Proceedings of the 21st Aerodynamic Decelerator Systems Technology Conference*. Dublin, Ireland: American Institute of Aeronautics and Astronautics, 23–26 May 2011, paper AIAA 2011-2531.

[50] H. Altmann, “Influence of wind on terminal approach and landing accuracy,” in *Proceedings of the 22nd AIAA Aerodynamic Decelerator Systems (ADS) Conference*. American Institute of Aeronautics and Astronautics, 25–28 Mar. 2013. [Online]. Available: http://dx.doi.org/10.2514/6.2013-1345

[51] M. Ward, C. Montalvo, and M. Costello, “Performance characteristics of an autonomous airdrop system in realistic wind environments,” presented at the AIAA Guidance, Navigation, and Control Conference. Toronto, Ontario, Canada:

American Institute of Aeronautics and Astronautics, 2–5 Aug. 2010, paper AIAA-2010-7510.

[52] A. J. Calise and D. Preston, "Approximate correction of guidance commands for winds," presented at the 20th AIAA Aerodynamic Decelerator Systems Technology Conference. Seattle, WA: American Institute of Aeronautics and Astronautics, 4–7 May 2009, paper AIAA-2009-2997.

[53] G. Brown, R. Haggard, and J. Fogleman, "Parafoils for shipboard recovery of uavs," presented at the 11th AIAA Aerodynamic Decelerator Systems Technology Conference, San Diego, CA, 9–11 Apr. 1991, paper AIAA-1991-0835, pp. 48–53.

[54] J. W. Woods, *Multidimensional Signal, Image, and Video Processing and Coding*. Burlington, MA: Elsevier, 2006.

[55] S. E. Umbaugh, *Computer Vision and Image Processing: A Practical Approach Using CVIPtools*. Upper Saddle River, NJ: Prentice-Hall, 1998.

[56] D. H. Ballard and C. M. Brown, *Computer Vision*. Englewood Cliffs, NJ: Prentice-Hall, 1982.

[57] Y. Wang, J. Ostermann, and Y.-Q. Zhang, *Video Processing and Communications*. Upper Saddle River, NJ: Prentice-Hall, 2002.

[58] J. Aggarwal and N. Nandhakumar, "On the computation of motion from sequences of images—a review," *Proceedings of the IEEE*, vol. 76, no. 8, pp. 917–935, Aug. 1988.

[59] S. Hutchinson, G. Hager, and P. Corke, "A tutorial on visual servo control," *IEEE Transactions on Robotics and Automation*, vol. 12, no. 5, pp. 651–670, Oct. 1996.

[60] F. Kendoul, "Survey of advances in guidance, navigation, and control of unmanned rotorcraft systems," *Journal of Field Robotics*, vol. 29, no. 2, pp. 315–378, Mar.–Apr. 2012. [Online]. Available: http://dx.doi.org/10.1002/rob.20414

[61] U.S. Navy. (2006, Jan.). Autonomous fire scout UAV lands on ship. [Online]. Available: http://www.news.navy.mil/search/display.asp?story_id=22038

[62] CybAero. CybAero performs unmanned aerial missions in the arctic ocean. [Online]. Available: http://www.cybaero.se

[63] M. Garratt, H. Pota, A. Lambert, S. Eckersley-Maslin, and C. Farabet, "Visual tracking and LIDAR relative positioning for automated launch and recovery of an unmanned rotorcraft from ships at sea," *Naval Engineers Journal*, vol. 121, no. 2, pp. 99–110, Jun. 2009. [Online]. Available: http://dx.doi.org/10.1111/j.1559-3584.2009.00194.x

[64] M. I. Lizarraga, "Autonomous landing system for a UAV," M.S. thesis, Naval Postgraduate School, Monterey, CA, Mar. 2004. [Online]. Available: http://edocs.nps.edu/npspubs/scholarly/theses/2004/Mar/04Mar%5FLizarraga.pdf

[65] I. Kaminer, O. Yakimenko, V. Dobrokhodov, and B.-A. Lim, "Development and flight testing of gnc algorithms using a rapid flight test prototyping system," in *Proceedings of the Guidance, Navigation, and Control Conference and Exhibit*. Monterey, CA: American Institute of Aeronautics and Astronautics, 5–8 Aug. 2002, paper AIAA-2002-4653.

[66] O. Yakimenko, I. Kaminer, W. Lentz, and P. Ghyzel, "Unmanned aircraft navigation for shipboard landing using infrared vision," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 38, no. 4, pp. 1181–1200, Oct. 2002. [Online]. Available: http://dx.doi.org/10.1109/TAES.2002.1145742

[67] L. Coutard, F. Chaumette, and J.-M. Pflimlin, "Automatic landing on aircraft carrier by visual servoing," in *Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on*. San Francisco, CA: Institute of Electrical and Electronics Engineers, 25–30 Sep. 2011, pp. 2843–2848.

[68] L. Coutard, "Appontage automatique d'avions asservissement visuel," Ph.D. dissertation, Université de Rennes, Rennes, Brittany, France, Dec. 2012.

[69] D. B. Barber, S. R. Griffiths, T. W. McLain, and R. W. Beard, "Autonomous landing of miniature aerial vehicles," *Journal of Aerospace Computing, Information, and Communication*, vol. 4, no. 5, pp. 770–784, Jul. 2007. [Online]. Available: http://arc.aiaa.org/doi/abs/10.2514/1.26502

[70] B. Barber, T. McLain, and B. Edwards, "Vision-based landing of fixed-wing miniature air vehicles," *Journal of Aerospace Computing, Information, and Communication*, vol. 6, pp. 207–226, Mar. 2009.

[71] A. A. Proctor and E. N. Johnson, "Vision-only aircraft flight control methods and test results," in *Proceedings of the Guidance, Navigation, and Control Conference and Exhibit*. Providence, RI: American Institute of Aeronautics and Astronautics, 16–19 Aug. 2004, paper AIAA 2004-5351.

[72] A. A. Proctor and E. N. Johnson, "Vision-only approach and landing," in *Proceedings of the Guidance, Navigation, and Control Conference and Exhibit*. San Francisco, CA: American Institute of Aeronautics and Astronautics, 15–18 Aug. 2005, paper AIAA 2005-5871.

[73] S. Nardone, A. G. Lindgren, and K. F. Gong, "Fundamental properties and performance of conventional bearings-only target motion analysis," *IEEE Transactions on Automatic Control*, vol. 29, no. 9, pp. 775–787, 1984.

[74] E. W. Frew, "Observer trajectory generation for target-motion estimation using monocular vision," Ph.D. dissertation, Stanford University, Aug. 2003.

[75] D. B. Barber, J. D. Redding, T. W. McLain, R. W. Beard, and C. N. Taylor, "Vision-based target geo-location using a fixed-wing miniature air vehicle," *Journal of Intelligent & Robotic Systems*, vol. 47, no. 4, pp. 361–382, Dec. 2006, copyright - Springer Science+Business Media B.V. 2006; Last updated - 2011-08-09. [Online]. Available: http://dx.doi.org/10.1007/s10846-006-9088-7

[76] D. L. Johansen, J. K. Hall, R. W. Beard, and C. N. Taylor, "Stabilization of video from miniature air vehicles for target localization," *Journal of Aerospace Computing, Information, and Communication*, vol. 5, no. 8, pp. 251–273, Jul. 2008. [Online]. Available: http://arc.aiaa.org/doi/abs/10.2514/1.34228

[77] E. N. Johnson, A. J. Calise, Y. Watanabe, J. Ha, and J. C. Neidhoefer, "Real-time vision-based relative aircraft navigation," *Journal of Aerospace Computing, Information, and Communication*, vol. 4, no. 4, pp. 707–738, Apr. 2007. [Online]. Available: http://arc.aiaa.org/doi/abs/10.2514/1.23410

[78] T. Oliveira and P. Encarnação, "Ground target tracking control system for unmanned aerial vehicles," *Journal of Intelligent & Robotic Systems*, vol. 69, no. 1-4, pp. 373–387, Jan. 2013. [Online]. Available: http://dx.doi.org/10.1007/s10846-012-9719-0

[79] I. Kaminer, W. Kang, O. Yakimenko, and A. Pascoal, "Application of nonlinear filtering to navigation system design using passive sensors," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 37, no. 1, pp. 158–172, Jan. 2001. [Online]. Available: http://dx.doi.org/10.1109/7.913675

[80] P. Oliveira, "Periodic and non-linear estimators with applications to the navigation of ocean vehicles," Ph.D. dissertation, Instituto Superior Técnico, Lisbon, Portugal, Jul. 2002.

[81] J. Hespanha, O. Yakimenko, I. Kaminer, and A. Pascoal, "Linear parametrically varying systems with brief instabilities: an application to vision/inertial navigation," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 40, no. 3, pp. 889–902, Jul. 2004. [Online]. Available: http://dx.doi.org/10.1109/TAES.2004.1337462

[82] V. N. Dobrokhodov, I. I. Kaminer, K. D. Jones, and R. Ghabcheloo, "Vision-based tracking and motion estimation for moving targets using unmanned air vehicles," *Journal of Guidance, Control, and Dynamics*, vol. 31, no. 4, pp. 907–917, Jul.–Aug. 2008. [Online]. Available: http://dx.doi.org/10.2514/1.33206

[83] R. J. Schalkoff, *Digital Image Processing and Computer Vision*. New York, NY: John Wiley & Sons, Inc., 1989.

[84] M. T. Watson, "Vision guidance controller for an unmanned aerial vehicle," M.S. Thesis, Naval Postgraduate School, Monterey, CA, Dec. 1998. [Online]. Available: http://edocs.nps.edu/npspubs/scholarly/theses/1998/Dec/98Dec_Watson.pdf

[85] P. A. Ghyzel, "Vision-based navigation for autonomous landing of unmanned aerial vehicles," M.S. thesis, Naval Postgraduate School, Monterey, CA, Sep. 2000. [Online]. Available: http://edocs.nps.edu/npspubs/scholarly/theses/2000/Sep/00Sep_Ghyzel.pdf

[86] L. V. Schmidt, *Introduction to Aircraft Flight Dynamics*, ser. AIAA Education Series. Reston, VA: American Institute of Aeronautics and Astronautics, 1998.

[87] A. Gelb, Ed., *Applied Optimal Estimation*. Cambridge, MA: The M.I.T. Press, 1974.

[88] P. S. Maybeck, *Stochastic Models, Estimation, and Control*, ser. Mathematics in Science and Engineering. Orlando, FL: Academic Press, Inc., 1979, vol. 141, no. 1.

[89] P. Zarchan and H. Musoff, *Fundamentals of Kalman Filtering: A Practical Approach*, ser. Progress in Aeronautics and Astronautics. Reston, VA: American Institute of Aeronautics and Astronautics, 2000, vol. 190.

[90] M. S. Grewal and A. P. Andrews, *Kalman Filtering*, 3rd ed. Hoboken, NJ: John Wiley & Sons, Inc., 2008.

[91] R. G. Brown and P. Y. C. Hwang, *Introduction to Random Signals and Applied Kalman Filtering*, 4th ed. Hoboken, NJ: John Wiley & Sons, Inc., 2012.

[92] C. Van Loan, "Computing integrals involving the matrix exponential," *IEEE Transactions on Automatic Control*, vol. 23, no. 3, pp. 395–404, Jun. 1978.

[93] T. Song, J. Ahn, and C. Park, "Suboptimal filter design with pseudomeasurements for target tracking," *IEEE Transactions on Aerospace and Electronic Systems*, vol. 24, no. 1, pp. 28–39, Jan. 1988.

[94] B. W. Parkinson and J. J. S. Jr., Eds., *The Global Positioning System: Theory and Applications, Volume 1*, ser. Progress in Astronautics and Aeronautics. Washington, DC: American Institute of Aeronautics and Astronautics, 1996, vol. 163.

[95] D. J. Biezad, *Integrated Navigation and Guidance Systems*, ser. AIAA Education Series. Reston, VA: American Institute of Aeronautics and Astronautics, 1999.

[96] F. Mohd-Yasin, D. J. Nagel, and C. E. Korman, "Noise in MEMS," *Measurement Science and Technology*, vol. 21, no. 1, 2010. [Online]. Available: http://stacks.iop.org/0957-0233/21/i=1/a=012001

[97] L. Ma, C. Cao, N. Hovakimyan, V. Dobrokhodov, and I. Kaminer, "Adaptive vision-based guidance law with guaranteed performance bounds," *Journal of Guidance, Control, and Dynamics*, vol. 33, no. 3, pp. 834–852, May–Jun. 2010. [Online]. Available: http://dx.doi.org/10.2514/1.46287

[98] Y. Ma, S. Soatto, J. Košecká, and S. S. Sastry, *An Invitation to 3-D Vision: From Images to Geometric Models*, ser. Interdisciplinary Applied Mathematics. New York, NY: Springer-Verlag, 2004.

[99] E. A. Wan and R. van der Merwe, *Kalman Filtering and Neural Networks*. New York, NY: John Wiley & Sons, Inc., 2001, ch. 7, pp. 221–280.

[100] R. M. Rogers, *Applied Mathematics in Integrated Navigation Systems*, ser. AIAA Education Series. Reston, VA: American Institute of Aeronautics and Astronautics, 2000.

[101] O. A. Yakimenko, *Engineering Computations and Modeling in MATLAB/Simulink*, ser. AIAA Education Series. Reston, VA: American Institute of Aeronautics and Astronautics, 2011.

[102] J. Petrich and K. Subbarao, "On-board wind speed estimation for UAVs," in *Proceedings of the Guidance, Navigation, and Control and Co-located Conferences*, American Institute of Aeronautics and Astronautics. Portland, OR: American Institute of Aeronautics and Astronautics, 8–11 Mar. 2011. [Online]. Available: http://dx.doi.org/10.2514/6.2011-6223

[103] M. Ward, M. Costello, and N. Slegers, "Specialized system identification for parafoil and payload systems," *Journal of Guidance, Control, and Dynamics*, vol. 35, no. 2, pp. 588–597, Mar.–Apr. 2012. [Online]. Available: http://dx.doi.org/10.2514/1.53364

[104] L. M. Corp., "Lockheed martin applies wind measurement technology for more precise cargo airdrops to u.s. ground forces," Press release, 12 Jun. 2014. [Online]. Available: http://www.lockheedmartin.com/us/news/press-releases/2014/june/0612-12-airdrop.html

[105] D. Lee, N. Sezer-Uzol, J. F. Horn, and L. N. Long, "Simulation of helicopter shipboard launch and recovery with time-accurate airwakes," *Journal of Aircraft*, vol. 42, no. 2, pp. 448–461, Mar.–Apr. 2005. [Online]. Available: http://dx.doi.org/10.2514/1.6786

[106] M. R. Snyder, J. P. Shishkoff, F. D. Roberson, M. C. McDonald, C. J. Brownell, L. Luznik, D. S. Miklosovic, J. S. Burks, H. S. Kang, and C. H. Wilkinson, "Comparison of experimental and computational ship air wakes for YP class patrol craft," in *Launch & Recovery Symposium*. Arlington, VA: American Society of Naval Engineers, 7–9 Dec. 2010.

[107] C. J. Brownell, L. Luznik, M. R. Snyder, H. S. Kang, and C. H. Wilkinson, "In situ velocity measurements in the near-wake of a ship superstructure," *Journal of Aircraft*, vol. 49, no. 5, pp. 1440–1450, Sep.–Oct. 2012. [Online]. Available: http://dx.doi.org/10.2514/1.C031727

[108] M. Snyder, H. Kang, and J. Burks, "Validation of computational ship air wakes for a naval research vessel," in *Proceedings of the 51st AIAA Aerospace Sciences Meeting including the New Horizons Forum and Aerospace Exposition*, American Institute of Aeronautics and Astronautics. Grapevine, TX: American Institute of Aeronautics and Astronautics, 7–10 Jan. 2013. [Online]. Available: http://dx.doi.org/10.2514/6.2013-959

[109] R. Corless, G. Gonnet, D. Hare, D. Jeffrey, and D. Knuth, "On the Lambert *W* function," *Advances in Computational Mathematics*, vol. 5, pp. 329–359, 1996. [Online]. Available: http://dx.doi.org/10.1007/BF02124750

[110] S. Haykin, *Adaptive Filter Theory*, 4th ed. Upper Saddle River, N.J.: Prentice Hall, 2002.

[111] D. Manolakis, V. Ingle, and S. Kogon, *Statistical and Adaptive Signal Processing: Spectral Estimation, Signal Modeling, Adaptive Filtering and Array Processing*. Artech House, Inc., 2005.

[112] P. A. Mortaloni, O. A. Yakimenko, V. N. Dobrokhodov, and R. M. Howard, "On the development of a six-degree-of-freedom model of a low-aspect-ratio parafoil

delivery system," presented at the 17th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar, Monterey, CA, 19–22 May 2003, paper AIAA-2003-2105.

[113] E. Bourakov, O. Yakimenko, and N. Slegers, "Exploiting a GSM network for precise payload delivery," in *Proceedings of the 20th AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar*. Seattle, WA: American Institute of Aeronautics and Astronautics, 4–7 May 2009, paper AIAA-2009-3004. [Online]. Available: http://dx.doi.org/10.2514/6.2009-3004

[114] A. Yingling, C. Hewgley, T. Seigenthaler, and O. Yakimenko, "Miniature autonomous rocket recovery system (MARRS)," in *Proceedings of the 21st AIAA Aerodynamic Decelerator Systems Technology Conference and Seminar*. Dublin, Ireland: American Institute of Aeronautics and Astronautics, 23–26 May 2011, paper AIAA-2011-2597. [Online]. Available: http://dx.doi.org/10.2514/6.2011-2597

[115] J. E. Benton and O. A. Yakimenko, "On development of autonomous HAHO parafoil system for targeted payload return," in *Proceedings of the 22nd AIAA Aerodynamic Decelerator Systems (ADS) Conference*. Daytona Beach, FL: American Institute of Aeronautics and Astronautics, 25–28 Mar. 2013, paper AIAA-2013-1312. [Online]. Available: http://dx.doi.org/10.2514/6.2013-1312

THIS PAGE INTENTIONALLY LEFT BLANK

Initial Distribution List

1. Defense Technical Information Center
   Ft. Belvoir, Virginia

2. Dudly Knox Library
   Naval Postgraduate School
   Monterey, California